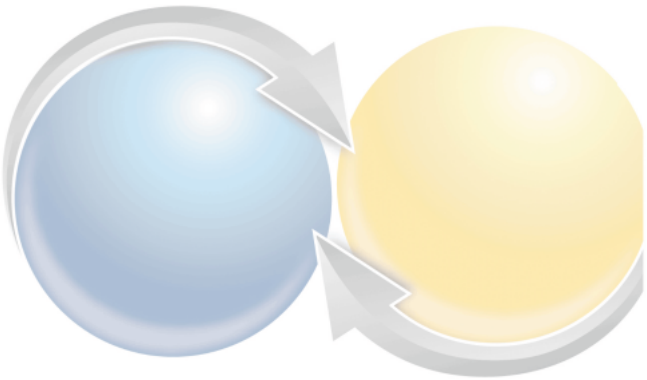
Livelink WCM Server

WCM WebServices Programmierhandbuch

Dieses Handbuch beschreibt die Web-Services-Programmierschnittstelle, mit der über das Internet auf Funktionen und Methoden von Livelink WCM Server zugegriffen werden kann. Sie erhalten Informationen zu folgenden Themen:

- grundlegende Konzepte von WCM WebServices
- Einsatz von WCM WebServices
- Zugriff auf Administrationsdaten und Authentifizierung sowie Objektverwaltung über WCM WebServices
- Anwendungsbeispiele



OPEN TEXT CORPORATION

Great Minds Working Together™

Copyright 2005 Open Text Corporation

Das Copyright an diesen Unterlagen und der dazugehörigen Software gehört ohne Einschränkungen Open Text. Diese Unterlagen und die dazugehörige Software dürfen ohne die ausdrückliche, schriftliche Genehmigung von Open Text weder ganz noch teilweise kopiert werden.

Die Open Text Corporation ist Eigentümer der Warenzeichen Open Text, 'Great Minds Working Together', Livelink, MeetingZone u.a.; diese Liste ist nicht vollständig. Andere erwähnte Warenzeichen sind Eigentum des jeweiligen Unternehmens und werden nur zum Zweck der Identifizierung der Produkte und Unternehmen verwendet. Alle Rechte vorbehalten.

Für die in diesem Dokument beschriebene Software der Open Text Corporation gelten bestimmte Gewährleistungen und Einschränkungen. Informationen zu diesen Gewährleistungen und Einschränkungen erhalten Sie in dem Lizenzvertrag, der zwischen dem Lizenznehmer und der Open Text Corporation geschlossen wurde.

Kontaktadresse:

Unternehmenshauptsitz
Open Text Corporation
185 Columbia Street West,
Waterloo, Ontario
N2L 5Z5
Kanada

Telefon: +1 519 888-7111

Wenn Sie Abonnent des Software Maintenance Program sind oder weitere Informationen zu anderen Kundendienstprogrammen wünschen, wenden Sie sich an den Kundendienst von Open Text unter http://www.opentext.com/services/support.html.

Wenn Sie zu dieser Veröffentlichung Vorschläge machen möchten, senden Sie eine E-Mail-Nachricht an documentation@opentext.com.

Weitere Informationen zu den Produkten und Dienstleistungen von Open Text finden Sie auf unserer Homepage unter http://www.opentext.com.

© 2005 IXOS SOFTWARE AG

Bretonischer Ring 12
85630 Grasbrunn, Deutschland

Tel.: +49 (89) 4629-0
Fax: +49 (89) 4629-1199
E-Mail: <office@ixos.de>
Internet: http://www.ixos.de

Alle Rechte vorbehalten. Einschließlich solche, die die Reproduktion, das Kopieren oder eine andere Verwendung oder Übermittlung der Inhalte dieses Dokumentes oder Teile davon betreffen. Kein Teil dieser Publikation darf, egal in welcher Form, ohne die schriftliche Zustimmung der IXOS SOFTWARE AG reproduziert, an Dritte übermittelt, unter Einsatz elektronischer Retrieval-Systeme verarbeitet, kopiert, verteilt oder für öffentliche

Vorführungen verwendet werden. IXOS behält sich das Recht vor, Aktualisierungen und Änderungen der Inhalte vorzunehmen. Sämtliche Daten, die auf Bildschirmphotographien (screenshots) sichtbar sind, dienen lediglich als Beispiel zur Demonstration der Software. Für den Inhalt dieser Daten übernimmt IXOS keine Gewähr. Dieses Produkt beinhaltet Software, die im Rahmen des Projekts OpenSSL für den Gebrauch im OpenSSL Toolkit (http://www.openssl.org/) bzw. die durch die Apache Software Foundation (http://www.apache.org/) entwickelt wurde.

Marken IXOS: IXOS SOFTWARE AG.

SAP®, R/3® und SAP ArchiveLink® sind eingetragene Marken der SAP AG.

Microsoft®, Microsoft Windows NT® und die Namen weiterer Microsoft-Produkte sind eingetragene Marken der Microsoft Corporation.

Acrobat Reader Copyright © 1987 Adobe Systems Incorporated. Alle Rechte vorbehalten. Adobe und Acrobat sind Marken von Adobe Systems Incorporated, die in bestimmten Rechtsbereichen registriert sein können.

Siebel® ist eingetragene Marke der Siebel Systems, Inc.

Sonstige Produktnamen werden nur zur Identifikation der Produkte verwendet und können eingetragene Marken der entsprechenden Hersteller sein.

Copyright © 2005 Gauss Interprise AG Hamburg, Gauss Interprise, Inc. Irvine, California. Alle Rechte weltweit vorbehalten.

Dieses Dokument sowie die zugehörige Software sind Eigentum der Gauss Interprise AG oder ihrer Zulieferer und durch Gesetze zum Schutze des Urheberrechts und andere Gesetze geschützt. Sie werden unter einer Lizenz vertrieben, durch welche die Nutzung, Reproduktion, Vertrieb und Dekompilierung eingeschränkt wird. Weder der Erhalt noch der Besitz dieses Dokumentes ermächtigt Sie, dessen Inhalte ganz oder teilweise auf Papier, elektronisch oder einem anderen Medium zu reproduzieren, weiterzugeben oder anderen den Zugang darauf zu ermöglichen. Kein Teil dieses Dokumentes darf in irgendeiner Form und Weise ohne vorherige schriftliche Zustimmung der Gauss Interprise AG oder Gauss Interprise, Inc. reproduziert werden.

Darüber hinaus gelten für diese Dokumentation die Bestimmungen des Softwarelizenzvertrags.

Alle Warenzeichen oder Handelsmarken, die in diesem Dokument erwähnt wurden, sind Eigentum der entsprechenden Firmen.

http://www.gaussvip.com

Programmversion: Livelink Web Content Management Server™ (Content Server) 9.5.0

Dokumentenversion: De-03

Erscheinungsdatum: März 2005

Inhaltsverzeichnis

Abbildungsverzeichnis **6**

Tabellenverzeichnis **7**

Kapitel 1 Einleitung **11**

1.1 Hinweise zu dieser Dokumentation 12

1.2 Typographische Konventionen 16

Kapitel 2 Konzepte **19**

2.1 Web-Services 19

2.2 WCM WebServices 26

2.3 Konzepte von Livelink WCM Server 28

Kapitel 3 Einsatz von WCM WebServices **39**

3.1 Konfiguration 39

3.2 Die WCM WebServices-Beschreibung 44

3.3 Fehlerbehandlung 47

Kapitel 4 Zugriff auf Administrationsdaten und Authentifizierung **53**

4.1 Benutzer, Gruppen und Rollen 54

4.2 Funktionsbereiche 62

4.3 Websites 65

4.4 Deploymentsysteme 66

4.5 Runlevels 67

4.6 Allgemeine Abfragen 70

4.7 Authentifizierung 71

Kapitel 5 Objektverwaltung 75

5.1 WCM-spezifische Datentypen 76

5.2 Allgemeine Parameter 98

5.3 Staging-Methoden 100

5.4 Workflow-Methoden 109

5.5 Methoden der Objektverwaltung 119

5.6 Suchen nach WCM-Objekten 147

Kapitel 6 Anwendungsbeispiele 155

6.1 VisualBasic for Applications 156

6.2 C# und ASP.NET 176

Glossar 195

Index 201

Abbildungsverzeichnis

Abb. 1 –	Rollen und Aktionen innerhalb der Web-Service-Architektur	21
Abb. 2 –	Protokoll-Stack für Web-Services	23
Abb. 3 –	Methodenaufruf eines Web-Service-Clients	25
Abb. 4 –	Grundlegende Funktionsweise von WCM WebServices	27
Abb. 5 –	Applications des Content-Servers	42
Abb. 6 –	Parameter der Application `WebServiceApplication`	43
Abb. 7 –	Der Datentyp `Filter` und davon abgeleitete spezielle Filtertypen	148
Abb. 8 –	Einbindung der Bibliotheken im VisualBasic Editor	163
Abb. 9 –	C#-Projekt einrichten	177
Abb. 10 –	WSDL als Webverweis hinzufügen	178
Abb. 11 –	Benutzerkennung und Passwort eingeben	179
Abb. 12 –	Automatisch erzeugte Klassen im Namensraum `VipWebServiceClient`	180
Abb. 13 –	Die Beispielanwendung (Liste aller PDF-Dateien)	181
Abb. 14 –	Maildialog der Beispielanwendung	182

Tabellenverzeichnis

Tabelle 1 – Verfügbare Funktionalität in den APIs von Livelink WCM Server 14
Tabelle 2 – Exceptions und mögliche Ursachen 49
Tabelle 3 – Datentypen für Benutzer 55
Tabelle 4 – Datentypen für Gruppen und Rollen 56
Tabelle 5 – Parameter der Methode `getGroupProfile` 58
Tabelle 6 – Parameter der Methode `getRoleProfile` 59
Tabelle 7 – Parameter der Methode `getUserProfile` 60
Tabelle 8 – Namen der Standard-Funktionsbereiche 62
Tabelle 9 – Parameter der Methode `getDeploymentSystems` 67
Tabelle 10 – Parameter der Methode `changePassword` 71
Tabelle 11 – Parameter der Methode `substituteLogin` 72
Tabelle 12 – Die Komponenten des Datentyps `ObjectId` 77
Tabelle 13 – Die Komponenten des Datentyps `ObjectData` 79
Tabelle 14 – Die Komponenten des Datentyps `ObjectType` 85
Tabelle 15 – Die einzelnen Objektstatus 86
Tabelle 16 – Die möglichen Übergänge zwischen den Objektstatus 87
Tabelle 17 – Die Komponenten des Datentyps `ObjectState` 88
Tabelle 18 – Die möglichen Zugriffsrechte für WCM-Objekte 89
Tabelle 19 – Die Komponenten des Datentyps `Permission` 89
Tabelle 20 – Die Komponenten des Datentyps `Acl` 90
Tabelle 21 – Die Komponenten des Datentyps `AclEntry` 91
Tabelle 22 – Die Komponenten des Datentyps `MultiImportPart` 92
Tabelle 23 – Die Komponenten des Datentyps `Workflow` 95
Tabelle 24 – Die Komponenten des Datentyps `Activity` 96
Tabelle 25 – Die Komponenten des Datentyps `Transition` 97
Tabelle 26 – Die Komponenten des Datentyps `WorkflowNavigationInfo` 97

Tabelle 27 – Die Komponenten des Datentyps DeploymentWaitInfo 99
Tabelle 28 – Die Komponenten des Datentyps EMailInfo 100
Tabelle 29 – Parameter der Methode checkIn 101
Tabelle 30 – Parameter der Methode checkOut 102
Tabelle 31 – Parameter der Methode directRelease 103
Tabelle 32 – Parameter der Methode reject 104
Tabelle 33 – Parameter der Methode release 105
Tabelle 34 – Parameter der Methode submit 106
Tabelle 35 – Parameter der Methode submitImmediately 107
Tabelle 36 – Parameter der Methode undoCheckOut 108
Tabelle 37 – Parameter der Methode assignWorkflow 109
Tabelle 38 – Parameter der Methode assignWorkflowAsync 111
Tabelle 39 – Parameter der Methode forward 112
Tabelle 40 – Parameter der Methode getAssignedJobs 113
Tabelle 41 – Parameter der Methode getAssignedWorkflow 113
Tabelle 42 – Parameter der Methode getWorkflow 115
Tabelle 43 – Parameter der Methode isForwardable 116
Tabelle 44 – Parameter der Methode isRemovable 116
Tabelle 45 – Parameter der Methode removeWorkflow 117
Tabelle 46 – Parameter der Methode removeWorkflowAsync 118
Tabelle 47 – Parameter der Methode addRemark 121
Tabelle 48 – Parameter der Methode change 122
Tabelle 49 – Parameter der Methode checkReferencesForDelete 123
Tabelle 50 – Parameter der Methode checkReferencesForRelease 124
Tabelle 51 – Parameter der Methode checkReferencesForSubmit 125
Tabelle 52 – Parameter der Methode convertContent 126
Tabelle 53 – Parameter der Methode copy 127
Tabelle 54 – Parameter der Methode create 128
Tabelle 55 – Parameter der Methode delete 129

Tabelle 56 – Parameter der Methode `depublishPage` 130
Tabelle 57 – Parameter der Methode `destroy` 131
Tabelle 58 – Parameter der Methode `generatePage` 132
Tabelle 59 – Parameter der Methode `get` 133
Tabelle 60 – Parameter der Methode `getCheckOutContent` 134
Tabelle 61 – Parameter der Methode `getChildren` 135
Tabelle 62 – Parameter der Methode `getContent` 136
Tabelle 63 – Parameter der Methode `getDeploymentJobs` 137
Tabelle 64 – Parameter der Methode `getExternalLinks` 138
Tabelle 65 – Parameter der Methode `getLastLogEntries` 139
Tabelle 66 – Parameter der Methode `getParent` 140
Tabelle 67 – Parameter der Methode `getVersionList` 141
Tabelle 68 – Parameter der Methode `move` 141
Tabelle 69 – Parameter der Methode `multiImport` 143
Tabelle 70 – Parameter der Methode `restoreVersion` 145
Tabelle 71 – Parameter der Methode `SortParentsFirst` 146
Tabelle 72 – Attributtypen und Filter 149
Tabelle 73 – Parameter der Methode `filter` 152
Tabelle 74 – Klassenmodule 163
Tabelle 75 – ASP.NET-Beispiel 182

KAPITEL 1

Einleitung

WCM WebServices ist eine Programmierschnittstelle, die den Zugriff auf viele von Livelink Web Content Management Server™ (kurz: Livelink WCM Server) angebotene Funktionen ermöglicht. Mithilfe von WCM WebServices können Sie von einem beliebigen Rechner aus auf einen Content-Server und damit auf die WCM-verwalteten Inhalte zugreifen. Der Zugriff erfolgt völlig plattformunabhängig. Der Datenaustausch mit dem WCM-System erfolgt über das XML-basierte Protokoll SOAP.

WCM WebServices bietet Ihnen folgende Funktionen:

- Entwicklung eigenständiger WCM-Client-Applikationen, die auf Ihre individuellen Anforderungen zugeschnitten sind
- Erstellen dynamischer Inhalte für Ihre Website mit der Möglichkeit, auf die Funktionalitäten zur Verwaltung der Website-Inhalte zuzugreifen. Zur Implementation der dynamischen Inhalte können Sie eine beliebige Programmiersprache wählen.
- Integration der Funktionalitäten zur Verwaltung der Website-Inhalte in Desktop- und Office-Werkzeuge, bei denen der Funktionsumfang mit gängigen Programmiersprachen erweitert werden kann

WCM WebServices basiert auf dem WCM Java API und arbeitet mit den gleichen oder sehr ähnlichen Methoden, um den Zugriff über das Internet zu ermöglichen.

Achtung! Das falsche Verwenden der in diesem Handbuch beschriebenen Programmierschnittstelle kann zu Fehlern im WCM-System bis hin zum Absturz des Systems und zu Datenverlusten führen. Durch eine falsche Programmierung können auch Probleme in den Bereichen Performance und Systemressourcen auftreten. Es ist deshalb wichtig, die entwickelte Software erfolgreich hinsichtlich Korrektheit, Stabilität, Robustheit und Performance zu testen, bevor sie in den produktiven Betrieb übernommen wird.

Für das korrekte Funktionieren der entwickelten Software kann keinerlei Garantie durch die Gauss Interprise AG übernommen werden. Gern unterstützen wir Sie beratend und entwickelnd durch den Bereich Gauss Professional Services, um so Probleme bereits in einer frühen Phase der Entwicklung zu verhindern.

1.1 Hinweise zu dieser Dokumentation

Diese Dokumentation richtet sich an Betreiber eines WCM-Systems, die den Zugriff auf die Daten des WCM-Systems über Web-Services einrichten möchten (Dienstanbieter), und Softwareentwickler, die über WCM WebServices auf die Funktionalitäten des WCM-Systems zugreifen möchten (Dienstnehmer).

Um über WCM WebServices auf WCM-Funktionalitäten zugreifen zu können, werden entsprechende Vorkenntnisse über die Arbeitsweise und Funktionen von Livelink WCM Server vorausgesetzt. Die Realisierung einer Client-Applikation zur Nutzung von WCM-Funktionen kann in verschiedenen Programmiersprachen erfolgen.

Der Inhalt dieses Handbuchs ist folgendermaßen gegliedert:

- Kapitel 2 “Konzepte”erläutert die grundlegenden Konzepte von WCM WebServices und Livelink WCM Server.
- Kapitel 3 “Einsatz von WCM WebServices” ist vorrangig für den Dienstanbieter bestimmt. Es beschreibt die erforderlichen Konfigurationsmaßnahmen im WCM-System für den Einsatz von WCM WebServices. Außerdem behandelt es die WCM WebServices-Beschreibung und die Meldungen in Fehlersituationen.
- Kapitel 4 “Zugriff auf Administrationsdaten und Authentifizierung” beschreibt die Methoden zum Auslesen von Gruppen, Rollen und Benutzern des WCM-Systems, zum Auslesen von Systeminformationen sowie zur Authentifizierung.
- Kapitel 5 “Objektverwaltung” beschreibt die WCM-Datentypen und Methoden zum Zugriff auf WCM-Objekte.
- Einige Programmierbeispiele für WCM WebServices finden Sie im Kapitel 6 “Anwendungsbeispiele”.

Die hier beschriebene Programmierschnittstelle ist Bestandteil von Livelink WCM Server. Zusätzlich zum vorliegenden Programmierhandbuch können Sie Informationen aus folgenden Quellen beziehen:

- WCM Java API-Programmierhandbuch: Dieses Handbuch enthält Informationen über Interfaces, Klassen und Methoden der Java-Programmierschnittstelle (WCM Java API), mit der die Funktionalität der WCM-Server über externe Programme genutzt werden kann.
- Content-Client-Benutzerhandbuch: Dieses Handbuch weist Sie ausführlich in alle Aufgaben der redaktionellen Pflege von WCM-verwalteten Websites ein.

- Livelink WCM Server-Administratorhandbuch: Dieses Handbuch beschreibt die Konfiguration und Administration von WCM-Systemen und enthält eine ausführlich Darstellung der technischen Konzepte von Livelink WCM Server.
- Livelink WCM Server-Installationshandbuch: Dieses Handbuch beschreibt die Installation des WCM-Systems und gibt Konfigurationshinweise für HTTP-Server und JSP-Engine.

Es gibt verschiedene flexible Programmierschnittstellen, mit denen die Kernfunktionalitäten von Livelink WCM Server über externe Programme genutzt werden können. Die folgende Tabelle beinhaltet eine Übersicht über die verschiedenen Schnittstellen und die jeweils verfügbare Funktionalität.

Tabelle 1 – Verfügbare Funktionalität in den APIs von Livelink WCM Server

Funktionalität des	WCM Java API	Remote-API	Portal Manager API	WCM WebServices
`AdminHandler`	✓	✓	✓ nur lesend über `VipUserBean`	
`AttributeHandler`	✓	✓	✓ über `VipWebsiteBean`	✓ über `VipWebService_Port`
`EventDispatcher`	✓	✓		

Funktionalität des	WCM Java API	Remote-API	Portal Manager API	WCM WebServices
MailHandler	✓		✓ über VipEMailBean	
ObjectHandler	✓	✓	✓ über VipObject HandlerBean	✓ über VipWebService_ Port
ObjectLoader	✓	✓		
SystemHandler	✓	✓		
WorkflowHandler	✓		✓ über VipWorkflow Bean	✓ über VipWebService_ Port
DeploymentHandler	✓	✓		
ContextHandler	✓	✓		
LivelinkAdmin Handler	✓		✓ über LivelinkUser Bean	
LivelinkObject Handler	✓		✓ über LivelinkObject Bean	

1.2 Typographische Konventionen

Programmelemente u.Ä. werden im Text folgendermaßen hervorgehoben:

Element	Schriftart oder Symbol	Beispiele
Programmoberfläche wie z.B. Menübefehle, Fenster, Dialoge, Feld- und Schaltflächen-bezeichnungen	*Menü → Eintrag*	*Datei → Anlegen*
Pfade zu Verzeichnissen, Namen von Dateien und Verzeichnissen	**Laufwerk:\Verzeichnis\ Dateiname**	**D:\WCM\ admin.bat**

Zitate aus Programmcode oder Konfigurationsdateien	`Code-Zitate`	`<head>`
		`<title>heading </title>`
		`</head>`

Variablen, d.h. Platzhalter für bestimmte Elemente	{Variable}	{WCM-Installationsverzeichnis}

Wichtige **Hinweise** und **Warnungen** stehen in grauen Kästen. Diese Informationen sollten Sie unbedingt lesen, um Fehler bei der Nutzung und Verwaltung von WCM-Systemen sowie Datenverluste zu vermeiden.

KAPITEL 2

Konzepte

Dieses Kapitel bietet Ihnen eine allgemeine Übersicht über Web-Services. Anschließend wird auf die Programmierschnittstelle WCM WebServices und auf die Konzepte von Livelink WCM Server eingegangen.

2.1 Web-Services

Web-Services sind ein XML-basierter Standard für Schnittstellen. Sie ermöglichen eine direkte Kommunikation zwischen verschiedenen Applikationen über das Internet.

Ein Web-Service kann von einer Person, aber auch von einem anderen Web-Service aufgerufen werden, der den angebotenen Dienst bzw. die angebotene Funktion nutzen möchte. Für den Aufruf des Dienstes spielt die zugrunde liegende Software-Infrastruktur keine Rolle. Die Schnittstellen sind in einem einheitlichen Standard definiert und gestaltet. Somit können Web-Services auf Basis plattformunabhängiger Technologien angebunden werden. Für geschäftliche Anwendungen bedeutet dies, dass die Frage nach der Integrationssoftware – der so genannten Middleware zwischen zwei proprietären Systemen – universell gelöst werden kann.

Web-Service-Architektur

Die Web-Service-Architektur basiert auf dem Zusammenspiel der folgenden drei Rollen:

- dem Anbieter eines Web-Service bzw. Dienstes
- einem Dienstverzeichnis
- dem Dienstnehmer

Diese Rollen sind durch bestimmte Aktionen wie Veröffentlichen, Erstellen einer Anfrage sowie Aufbauen einer Verbindung miteinander verknüpft.

In einem typischen Web-Service-Szenario verfügt der Anbieter des Web-Service über einen bestimmten Dienst, den er über das Internet zugänglich machen möchte. Der Dienstanbieter definiert zu diesem Zweck eine formale Beschreibung seines Web-Service. Diese Beschreibung benötigt ein Dienstnehmer für den Zugriff auf den Web-Service. Die Beschreibung kann der Dienstanbieter in einem universellen Dienstverzeichnis veröffentlichen, damit sein Web-Service leicht gefunden werden kann.

Wenn der Dienstnehmer auf einen Web-Service zugreifen möchte, sucht er den Dienst bzw. seine Beschreibung in der Regel in dem Dienstverzeichnis. Die Dienstbeschreibung kann er aber auch direkt vom Dienstanbieter erhalten. Mithilfe dieser Beschreibung baut der Dienstnehmer eine Verbindung zum Dienstanbieter bzw. zu dessen Web-Service auf und kann so auf den gewünschten Dienst zugreifen.

Die folgende Abbildung veranschaulicht das Zusammenspiel zwischen den genannten Rollen und den einzelnen Aktionen.

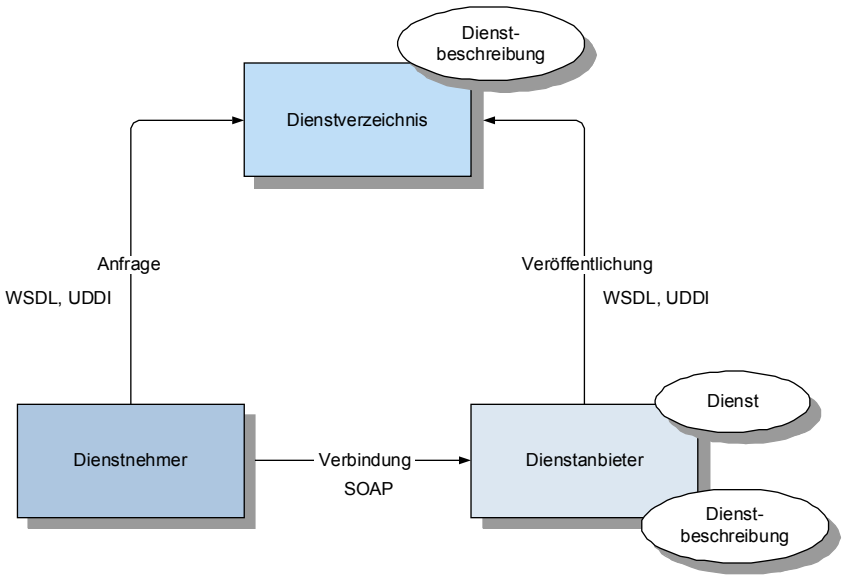


Abb. 1 – Rollen und Aktionen innerhalb der Web-Service-Architektur

Standards für Web-Services

Die folgenden Standards bestimmen im Wesentlichen den Web-Services-Standard:

- SOAP
- WSDL (Web Services Description Language)
- UDDI (Universal Description, Discovery, and Integration)

Diese Standards werden in den folgenden Abschnitten kurz beschrieben.

SOAP

SOAP ist ein Standard, der den plattformunabhängigen Zugriff auf Web-Services ermöglicht bzw. die Datenübertragung zwischen dem Dienstanbieter und dem Dienstnehmer definiert. Das verwendete Austauschformat ist XML.

WSDL

WSDL (Web Services Description Language) ist die Beschreibungssprache für Web-Services. Es handelt sich dabei um einen XML-basierten Standard.

Für die Nutzung eines Web-Service seitens einer Applikation ist eine genaue “Sprachregelung” bzw. Beschreibung erforderlich. Die Beschreibung muss genaue Informationen liefern über das verwendete Protokoll, die Adresse und Port-Nummer, die möglichen Prozeduren und Funktionen sowie die Formate für Eingabe und Ausgabe. Der Anbieter eines Web-Service stellt diese Informationen in Form einer WSDL-Datei zur Verfügung.

UDDI

UDDI (Universal Description, Discovery, and Integration) ist ebenfalls ein XML-basierter Standard. Er legt fest, wie Detailinformationen zu Web-Services und deren Anbietern in einheitlicher Form in Verzeichnissen abgelegt werden können.

Um die vielen verschiedenen Web-Services lokalisieren zu können, gibt es UDDI-Verzeichnisse, in denen Anbieter ihre Dienste registrieren und Dienstnehmer gezielt nach Web-Services suchen können. So entspricht ein UDDI-Verzeichnis einer Art (globaler) Gelber Seiten. Die Web-Services sind nach bestimmten Eigenschaften sortiert. Die entsprechenden Einträge referenzieren die WSDL-Dateien der zugehörigen Web-Services.

In einem UDDI-Verzeichnis sind Web-Services meist abhängig von geschäftlichen Anwendungen. Dementsprechend enthält das Verzeichnis eine Kategorisierung von Branchen und Unternehmen.

Das Web-Service-Schichtenmodell

Die folgende Abbildung veranschaulicht die Schichtung der einzelnen Protokolle.

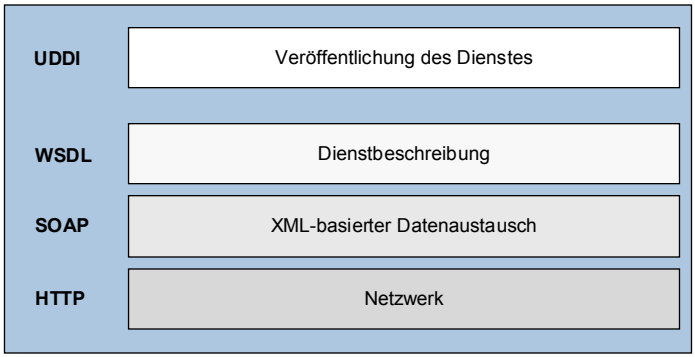


Abb. 2 – Protokoll-Stack für Web-Services

In der untersten Schicht des Protokoll-Stacks findet der Datentransport über das Standard-Protokoll HTTP statt. Oberhalb dieser Schicht erfolgt der XML-basierte Datenaustausch über den Standard SOAP. Der Standard beschreibt den Web-Service. Innerhalb der obersten Schicht wird der UDDI-Standard zur Veröffentlichung des Dienstes in einem UDDI-Verzeichnis verwendet. Diese Schicht ist optional. UDDI ist für die Funktionalität von Web-Services nicht zwingend erforderlich.

Web-Services-Workflow

Die folgenden Schritte fassen die Bereitstellung und Nutzung eines Web-Service inklusive der jeweils verwendeten Standards zusammen.

1. Der Dienstanbieter erstellt einen Web-Service.
2. Er liefert eine Beschreibung des Web-Service und verwendet dazu den Standard WSDL.

3. Der Dienstanbieter lässt den Web-Service in einem UDDI-Verzeichnis registrieren. Dieser Vorgang ist optional.
4. Ein anderer Web-Service oder Benutzer ermittelt den registrierten Web-Service über eine Suche in einem UDDI-Verzeichnis, das auch die erforderliche Beschreibung liefert. Alternativ kann die Beschreibung dem Dienstnehmer auch direkt vom Dienstanbieter zur Verfügung gestellt werden.

   Anschließend führt der Dienstnehmer einen Request an den ermittelten Web-Service durch.
5. Der Web-Service bzw. Benutzer, der den Request durchführt, verwendet eine Applikation, um eine Verbindung zu dem entsprechenden Web-Service aufzubauen. SOAP ist das eingesetzte Standard-Protokoll für den Zugriff auf den Web-Service.
6. Der eigentliche Datenaustausch erfolgt über das Austauschformat XML bzw. das Protokoll HTTP.

Web-Services nutzen

Die WSDL-Datei liefert die Beschreibung eines Web-Service. Zur Interpretation dieser Beschreibung stehen Dienstnehmern verschiedene Werkzeuge (Toolkits) zur Verfügung, die die Beschreibung einlesen und automatisch in ein für ihre Client-Anwendung nutzbares Modul umwandeln. Der Dienstnehmer – oder auch Client – bindet dieses Modul in seine Entwicklungsumgebung ein und schreibt seine eigene Applikation in der von seinem Werkzeug unterstützten Programmiersprache. Das Modul arbeitet wie ein Web-Service-Proxy, der Objekte annimmt und im Namen einer anderen Anwendung ausführt. Der Web-Service-Client kann also mithilfe des Moduls bzw. des Proxys den eigentlichen Web-Service ansprechen und dessen angebotenen Funktionen bzw. Methoden nutzen. Die entsprechenden Objekte werden über das Modul wieder an die Applikation gegeben.

Hinweis: Dienstnehmer, die mit SOAP vertraut sind, können die für den Zugriff auf den Web-Service erforderlichen HTTP-Requests auch ohne zusätzliches Werkzeug erstellen.

Die folgende Abbildung veranschaulicht das Zusammenspiel der einzelnen Komponenten.

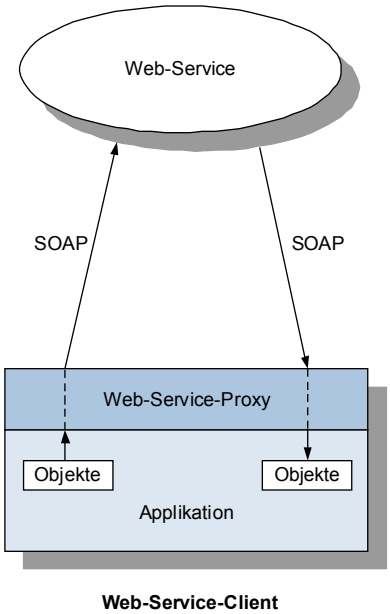


Abb. 3 – Methodenaufruf eines Web-Service-Clients

2.2 WCM WebServices

WCM WebServices ist eine Programmierschnittstelle, die den Zugriff auf Funktionen bzw. Methoden von Livelink WCM Server ermöglicht. Mithilfe von WCM WebServices kann von einem beliebigen Rechner über das Internet auf einen Content-Server und damit auf die WCM-verwalteten Inhalte zugegriffen werden. Die auf diese Weise verfügbaren Methoden können in einer beliebigen Programmiersprache aufgerufen werden.

WCM WebServices bietet Ihnen folgende Leistungsmerkmale:

- Unterstützung aller innerhalb von Livelink WCM Server verwendeten Datentypen
- lesender und schreibender Zugriff auf den Inhalt der WCM-Objekte
- Unterstützung des Staging-Konzepts von Livelink WCM Server
- Ausführen komplexer Suchanfragen
- Bereitstellung von Content Workflow, mit dem das dreistufige Staging-Konzept um eigene Workflow-Schritte erweitert werden kann

Der Content-Server, der im Kontext einer JSP-Engine bzw. als Webanwendung in einem Application-Server läuft, stellt für **jede** Website eine WSDL-Datei bereit. Diese enthält alle unterstützten Datentypen und Methoden. Pro Website kann es individuell definierte Attributmengen geben. Aus diesem Grund muss für jede Website eine WSDL-Datei zur Verfügung stehen.

Das für den Nachrichtenaustausch verwendete Protokoll SOAP besteht aus folgenden Komponenten:

- **SOAP-Envelope:** eine Art “Umschlag”, in dem beschrieben ist, was die Nachricht enthält und was damit geschehen soll
- **SOAP-Body:** enthält die eigentlichen Nutzdaten
- weitere Informationen zum Transport und Code

Die folgende Abbildung veranschaulicht das grundsätzliche Prinzip von WCM WebServices.

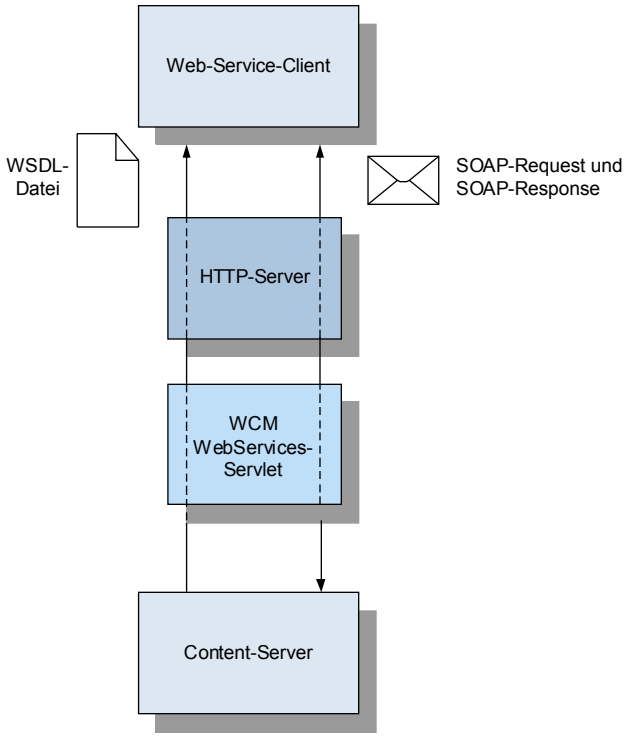


Abb. 4 – Grundlegende Funktionsweise von WCM WebServices

Der Web-Service-Client schickt eine Anfrage (SOAP-Request) in einem SOAP-Envelope an den Content-Server. Der SOAP-Envelope wird dabei vom WCM WebServices-Servlet zerlegt und der Inhalt der Nachricht (SOAP-Body) an den WCM WebServices-Handler weitergeleitet. Nach Bearbeitung der Nachricht durch den Content-Server wird eine entsprechende Antwort (SOAP-Response) mit ähnlichem Aufbau vom Servlet zusammengesetzt und an den Client zurückgeschickt. Das Modul des Clients empfängt die Antwort, trennt die SOAP-Bestandteile und stellt die Nutzdaten als Rückgabewerte der aufgerufenen Funktion zur Verfügung.

Hinweis: Der Zugriff auf den Content-Server über das WCM WebServices-Servlet erfordert eine Authentifizierung (Basic Authentication) des jeweiligen Benutzers. Dieser Benutzer muss im WCM-System angelegt sein.

2.3 Konzepte von Livelink WCM Server

Der folgende Abschnitt beschreibt in kurzer Form die grundlegenden Konzepte von Livelink WCM Server. Dabei steht die Beschreibung der Begriffe im Vordergrund, die beim Arbeiten mit WCM WebServices eine Rolle spielen.

Eine ausführliche Beschreibung der technischen Konzepte und der Architektur von Livelink WCM Server finden Sie im Livelink WCM Server-Administratorhandbuch. Weitere Informationen zum Arbeiten mit einem WCM-System sind im Content-Client-Benutzerhandbuch enthalten.

Objektdaten

Livelink WCM Server dient der Verwaltung von Websites. Websites enthalten bestimmte Dokumente, die für die verschiedensten Zielgruppen verwaltet werden. Solche Dokumente können z.B. HTML- oder XML-Dokumente, Grafiken oder Microsoft Word-Dokumente sein.

Die Dokumente einer von Livelink WCM Server verwalteten Website werden als WCM-Objekte bezeichnet. Jedes WCM-Objekt gehört zu einem bestimmten Typ (siehe Abschnitt “Objekttyp” auf Seite 31) und setzt sich aus folgenden Bestandteilen, den so genannten Objektdaten, zusammen:

- **Inhalt:** Der Inhalt (Content) eines WCM-Objekts sind die Daten des eigentlichen Dokuments wie z.B. die Zeichenfolge in einer HTML-Datei oder die Folge von Bytes einer Grafikdatei.
- **Metadaten:** Die Metadaten eines Objekts liefern Informationen über das Dokument, z.B. von wem das Dokument erstellt wurde, zu welchem Zeitpunkt es erstellt wurde, welchen Titel es hat, von welchem Typ es ist usw.
- **Protokoll:** Das Protokoll liefert eine Beschreibung sämtlicher Änderungen, die am Objekt durchgeführt wurden.

Sämtliche WCM-Objekte werden über ihre OID (Object Identifier) referenziert. Die OID identifiziert das *aktuelle* Objekt.

Neben einem definierten Satz von Metadaten können Sie jedem Objekttyp eine Attributmenge zuordnen. Dies ist eine Menge von Attributen, die spezielle Eigenschaften von Objekttypen beschreiben. Dazu können u.a. die Auflösung von Grafiken oder ein Copyright-Vermerk gehören.

Außerdem können Sie die Inhalte von WCM-Objekten kategorisieren. Zu diesem Zweck definieren Sie Objektkategorien, für die Sie eine Reihe von Eigenschaften festlegen können. Eine mögliche Objektkategorie wäre “Rechnung”, die durch die Eigenschaften “Rechnungsempfaenger” und “Status” gekennzeichnet sein können.

Objektstatus

Alle WCM-Objekte einer Website, die mit Livelink WCM Server verwaltet werden, durchlaufen feste Stufen aus Bearbeitung (Editieren), Qualitätssicherung und Veröffentlichung im Produktionsbetrieb. Durch das Zuordnen von Workflows zu WCM-Objekten lässt sich dieses so genannte *Staging* erweitern. So kann z.B. eine aufeinander folgende Bearbeitung durch mehrere Redakteure oder eine mehrstufige Qualitätssicherung durchgeführt werden.

Das Staging von Livelink WCM Server bestimmt den Status eines WCM-Objekts. Entsprechend den Aktionen der Objektverarbeitung werden die folgenden Objektstatus unterschieden:

- geändert (auch bei neu angelegten Objekten)
- ausgeliehen
- vorgelegt
- abgelehnt
- freigegeben
- verzögert freigegeben
- gelöscht

Durch die verschiedenen Arbeitsschritte bzw. Aktionen am WCM-Objekt ändert sich der jeweilige Status eines Objekts. Jede Statusänderung wird protokolliert. Dabei wird die vorherige Version eines WCM-Objekts immer in einer separaten Datenhaltung archiviert, sodass ältere Versionen nicht

verloren gehen. Auf diese Versionen können Sie jederzeit wieder zugreifen.

Welche Aktionen Sie an einem WCM-Objekt durchführen können, hängt von folgenden Faktoren ab:

- dem aktuellen Status des WCM-Objekts

  Hinweis: Die Möglichkeit, bestimmte Aktionen am Objekt durchzuführen, hängt auch vom Status des übergeordneten Themas innerhalb der Navigationstopologie ab.

- der Zugriffssteuerungsliste des WCM-Objekts
- den zugewiesenen Funktionsbereichen des Benutzers

Objekttyp

Der Objekttyp bestimmt, zu welcher Klasse von Dokumenten ein WCM-Objekt gehört. Daraus ergeben sich verschiedene Eigenschaften des WCM-Objekts, u.a.:

- Attribute aus einer Attributmenge, über die das WCM-Objekt verfügt
- Art der Vorlagen, die das WCM-Objekt verwenden kann
- Dateien, die bei der Seitengenerierung als Repräsentation des WCM-Objekts erzeugt werden

Neben Objekttypen wie z.B. “HTML-Seite”, “JSP-Vorlage” oder “PDF-Dokument” gibt es Themenobjekte. Ein Themenobjekt kann andere untergeordnete Objekte aufnehmen. Es dient insofern auch der Strukturierung der WCM-Objekte in der Navigationssicht und damit der Strukturierung der Website. Ein weiterer wichtiger Objekttyp ist die Vorlage. Sie bestimmt das Layout und ist für die Seitengenerierung wesentlich.

Seitengenerierung

Der Inhalt eines WCM-Objekts unterscheidet sich von den Daten, die für einen HTTP-Server generiert werden. Jedes WCM-Objekt kann eine Vorlage (Template) haben, die in der Regel ein Layout für die zu generierende Webseite definiert. Eine Vorlage ist selbst ein WCM-Objekt, das wiederum eine Vorlage enthalten kann. Die generierte Seite hängt u.a. von dieser Vorlagenhierarchie (Kaskade) ab.

Weiterhin spielen der Objekttyp des zu generierenden WCM-Objekts und die Objekttypen der Vorlagen eine wesentliche Rolle für die Seitengenerierung. So werden beispielsweise für GIF-Objekte zwei Dateien erzeugt: die Grafik selbst und eine HTML-Seite mit einem Verweis auf die Grafikdatei. Bei HTML-Objekten mit zugeordneter Vorlage wird der Body-Bereich des HTML-Dokuments ausgeschnitten.

Entscheidend ist auch die Datenhaltungssicht (Edit, QS oder Produktion), die bestimmt, welche Ausprägungen der WCM-Objekte für die Generierung verwendet werden, siehe dazu “Datenhaltungssichten” auf Seite 34.

Für die Seitengenerierung sind die *Deploymentsysteme* zuständig. Die Deploymentsysteme erzeugen aus den in der Datenbank gespeicherten WCM-Objekten Seiten, die mithilfe eines HTTP-Servers in einem Browser dargestellt werden können. Ein Deploymentsystem generiert die Seiten für eine Datenhaltungssicht und eine Website. In einem WCM-System können Sie beliebig viele Deploymentsysteme konfigurieren.

Jede Seite wird über eine URL referenziert. Da für ein und dasselbe WCM-Objekt von mehreren Deploymentsystemen verschiedene Seiten erzeugt werden können, gibt es im Allgemeinen mehrere URLs für ein und dasselbe WCM-Objekt.

Topologien – Organisation von WCM-Objekten

Die WCM-Objekte einer Website können aufgrund bestimmter Eigenschaften (Metadaten) als hierarchische Struktur mit über- und untergeordneten Objekten dargestellt werden. In einer Website gibt es zwei hierarchische Ordnungsmerkmale von WCM-Objekten: die Themenstruktur (Navigationstopologie) und die Vorlagenstruktur (Vorlagentopologie).

Themenstruktur (Navigationstopologie)

In der Themenstruktur werden Objekte Themen zugewiesen. Jedes Thema ist wiederum ein WCM-Objekt, das einem anderen Thema untergeordnet ist. Diese Hierarchie ist vergleichbar mit Verzeichnissen und Unterverzeichnissen in einem Dateisystem.

Für jede Website gibt es genau ein Thema, das das Wurzelobjekt des Navigationsbaums einer Website ist. Ausgehend vom Wurzelobjekt verzweigt der Baum in Themen und Unterthemen. Alle Objekte sind diesem Wurzelobjekt direkt oder indirekt untergeordnet.

Vorlagenstruktur (Vorlagentopologie)

In der Vorlagenstruktur werden Objekte nach ihren Vorlagen geordnet. Vorlagen sind ebenfalls WCM-Objekte. Jede Website kann eine oder mehrere Vorlagen enthalten, die ineinander verschachtelt sein können. Alle Objekte werden ihren jeweiligen Vorlagen untergeordnet. Objekte, die keine Vorlagen besitzen und selbst nicht vom Objekttyp “Vorlage” sind, werden in dieser Sicht nicht dargestellt.

Datenhaltungssichten

Die Erstellung und Bearbeitung von Objekten mit Livelink WCM Server unterliegen festen Stufen aus Editieren (Bearbeiten), Qualitätssicherung und Veröffentlichung im Produktionsbetrieb. Entsprechend dem Bearbeitungsstatus eines WCM-Objekts gibt es verschiedene Sichten auf das Objekt: Edit-, QS- und Produktionssicht.

- **Edit-Sicht:** Die Edit-Sicht repräsentiert den Arbeitsschritt der Erstellung und Bearbeitung von WCM-Objekten. Die Objekte wie beispielsweise HTML-Seiten werden von Redakteuren oder Grafikern angelegt und geändert. Dafür benötigen diese Benutzer entsprechende Zugriffsrechte für die Objekte. Nach der Bearbeitung werden die Objekte der Qualitätssicherung vorgelegt. Dadurch werden neu angelegte Objekte bzw. Änderungen an bestehenden Objekten auch in der QS-Sicht sichtbar.
- **QS-Sicht:** Die QS-Sicht zeigt die WCM-Objekte einschließlich aller Änderungen, die der Qualitätssicherung vorgelegt worden sind. Mitarbeiter der Qualitätssicherung können die Änderungen inhaltlich und formal prüfen. Aufgrund dieser Prüfung wird entschieden, ob ein Objekt zur Nachbesserung zurückgeschickt oder freigegeben wird. Für die Freigabe ist das entsprechende Zugriffsrecht nötig. Durch die Freigabe werden die Objekte in die Produktionssicht übertragen. Damit wird die aktuelle Version in der publizierten Website verfügbar.
- **Produktionssicht**: Die Produktionssicht stellt die freigegebenen Seiten einer Website bereit. Mithilfe eines HTTP-Servers kann auf die Seiten über das Internet oder Intranet zugegriffen werden.

Je nach Objektstatus sind die WCM-Objekte einer Website in verschiedenen Sichten verfügbar und können bearbeitet werden. Ein neu angelegtes Objekt ist beispielsweise nur in der Edit-Sicht sichtbar, nicht aber in der QS- oder Produktionssicht. Durch verschiedene Aktionen im Rahmen der Objektverarbeitung kann der Status eines WCM-Objekts in den verschiedenen Sichten unterschiedlich sein.

Authentifizierung und Zugriffsrechte

Für das Arbeiten mit dem WCM-System muss sich ein Benutzer authentifizieren. Bei der Anmeldung ist daher die Angabe einer eindeutigen Benutzerkennung (*User-ID*) und eines Passworts erforderlich. Die Überprüfung der Angaben übernimmt der Master-Administrationsserver des WCM-Systems.

Innerhalb eines WCM-Systems gibt es verschiedene Berechtigungen. Für die Arbeit mit WCM-Objekten sind vor allem die Objekt-Zugriffsrechte und Funktionsbereiche wichtig.

Zugriffsrechte für WCM-Objekte

Livelink WCM Server führt für jedes WCM-Objekt eine Zugriffssteuerungsliste (*Access Control List*, *ACL*). In dieser Liste ist festgelegt, über welche Zugriffsrechte die jeweiligen Benutzer, Gruppen oder Rollen (die so genannten *Principals*) verfügen. Dabei können Zugriffsrechte explizit als erlaubt bzw. verboten ausgezeichnet werden, oder sie werden nicht explizit gesetzt. Bei einer Rechteüberprüfung werden im letzteren Fall alle zugewiesenen Gruppen und Rollen dahingehend untersucht, ob sie eine explizite Erlaubnis oder ein Verbot definieren (wobei ein Verbot immer Vorrang hat).

Die Zugriffssteuerungslisten werden vererbt, d.h., jedes in der Navigationssicht tiefer angeordnete WCM-Objekt erbt in der Regel die Rechte des übergeordneten Objekts. Beim Anlegen einer Website wird ein Principal ausgewählt, der initial vollen Zugriff auf alle Objekte der Website hat (“Website-Administrator”). Dieser wird in der Zugriffssteuerungsliste des Wurzelobjekts der Website aufgenommen. Alle weiteren WCM-Objekte dieser Website erben, falls nicht explizit gesetzt, die Rechte des Wurzelobjekts.

Funktionsbereiche

Jedem Principal können Funktionsbereiche zugewiesen werden. Über die Funktionsbereiche steuern Sie, welche Funktionen von Livelink WCM Server dem Benutzer zur Verfügung stehen. Dabei erfüllen die Funktionsbereiche zwei wesentliche Aufgaben:

- Sie legen fest, welche Typen von Objekten die Benutzer anlegen, ausleihen und zurückgeben dürfen. Einige Funktionsbereiche wie z.B. “Basis” sind mit Objekttypen verknüpft. Nur Benutzer, die über den entsprechenden Funktionsbereich verfügen, können Objekte dieses Objekttyps anlegen, ausleihen und zurückgeben.
- Sie bestimmen, welche Ansichten und Dialoge dem Benutzer im Content-Client zur Verfügung stehen. Auf diese Weise können Sie genau festlegen, welche Aktionen der Benutzer durchführen kann.

Administrationsrechte für das WCM-System

Neben den Zugriffsrechten für WCM-Objekte gibt es außerdem Administrationsrechte für das WCM-System. Principals mit Administrationsrechten haben Zugriff auf den Admin-Client und können die Elemente des WCM-Systems wie Server, Websites, Deploymentsysteme und Benutzerinformationen bearbeiten. Die Administrationsrechte können abgestuft gewährt werden, sodass ein Benutzer nur Teilbereiche der Administration einsehen oder bearbeiten kann. Die Administrationsrechte können ausschließlich über den Admin-Client geändert werden.

KAPITEL 3

Einsatz von WCM WebServices

Diese Kapitel beschreibt die erforderliche Konfiguration für den Einsatz von WCM WebServices. Außerdem behandelt es die WCM WebServices-Beschreibung und die Meldungen in Fehlersituationen.

3.1 Konfiguration

Damit Benutzer über WCM WebServices auf Ihr WCM-System zugreifen können, sind folgende Konfigurationsmaßnahmen erforderlich:

- Einrichten eines Servlet-Mappings
- Anpassungen in der Konfiguration von Livelink WCM Server

Hinweise:

Wenn Sie zum Einbinden des Content-Servers in einen Application-Server eine Webanwendung (WAR-Datei) über den Admin-Client erzeugen, dann erfolgt die notwendige Konfiguration automatisch.

Falls Sie mit dem MS Internet Information Server arbeiten, schalten Sie aus Performance-Gründen den Cache für die ISAPI-Applikationen für die Website ein.

Servlet-Mapping einrichten

Richten Sie ein Servlet-Mapping für das WCM WebServices-Servlet (Klasse `de.gauss.vip.webservice.transport.WebServiceServlet`) in der Konfiguration des Application-Servers bzw. der JSP-Engine ein.

Das Mapping für Application-Server, die die Servlet-Spezifikation 2.2 erfüllen, erfordert folgende Einträge in der Datei **web.xml**. Diese Datei befindet sich im Verzeichnis **\WEB-INF** der Webanwendung, in der der Content-Server gestartet wird.

```
<servlet>
      <servlet-name>WebServiceServlet</servlet-name>
      <display-name>WCM WebServices-Servlet</display-name>
      <servlet-class>
         de.gauss.vip.webservice.transport.WebServiceServlet
      </servlet-class>
</servlet>
...
<servlet-mapping>
      <servlet-name>WebServiceServlet</servlet-name>
      <url-pattern>/WebService/*</url-pattern>
</servlet-mapping>
```

Sämtliche Anfragen, die mit `/WebService/` beginnen, werden an das WCM WebServices-Servlet weitergeleitet.

Besonderheiten von Resin

Wenn Sie Resin als JSP-Engine verwenden, sind zusätzlich folgende Schritte erforderlich:

1. Setzen Sie in der Datei **{Resin-Installationsverzeichnis}\conf\resin.conf** folgende System-Properties:

```
<system-property javax.xml.parsers.DocumentBuilderFactory=
   "org.apache.xerces.jaxp.DocumentBuilderFactoryImpl"/>
<system-property javax.xml.parsers.SAXParserFactory=
   "org.apache.xerces.jaxp.SAXParserFactoryImpl"/>

<!-- xslt -->

<system-property javax.xml.transform.TransformerFactory=
   "org.apache.xalan.processor.TransformerFactoryImpl"/>
```

2. Kopieren Sie folgende JAR-Dateien aus dem Verzeichnis **{WCM-Installationsverzeichnis}\lib** in das Verzeichnis **{Resin-Installationsverzeichnis}\lib**:
   - **xalan.jar**
   - **xercesImpl.jar**
   - **xml-apis.jar**
   - **xmlParserAPIs.jar**

Anpassungen in der Konfiguration durchführen

Damit Benutzer über WCM WebServices auf den Content-Server zugreifen können, sind folgende Anpassungen in der Konfiguration erforderlich:

- Zuordnen der Application `WebServiceApplication` zum Content-Server
- Konfigurieren der Application `WebServiceApplication` mithilfe von Parametern

Die entsprechenden Einträge für die Konfiguration nehmen Sie über den Admin-Client vor.

Hinweise:

Bitte beachten Sie, dass der Content-Server im Kontext einer JSP-Engine bzw. als Webanwendung in einem Application-Server laufen muss.

Genaue Informationen darüber, wie Sie einem Content-Server Applications zuordnen und die entsprechenden Parameter festlegen, finden Sie im Portal Manager API-Programmierhandbuch.

Application `WebServiceApplication` zuordnen

Für den Einsatz von WCM WebServices ordnen Sie die Application `WebServiceApplication` dem entsprechenden Content-Server zu. Starten Sie dazu den Admin-Client von Livelink WCM Server. Nehmen Sie die Zuordnung in der Ansicht *Konfiguration* über *Server* → *{Name des Content-Servers}* → *Applications* vor.

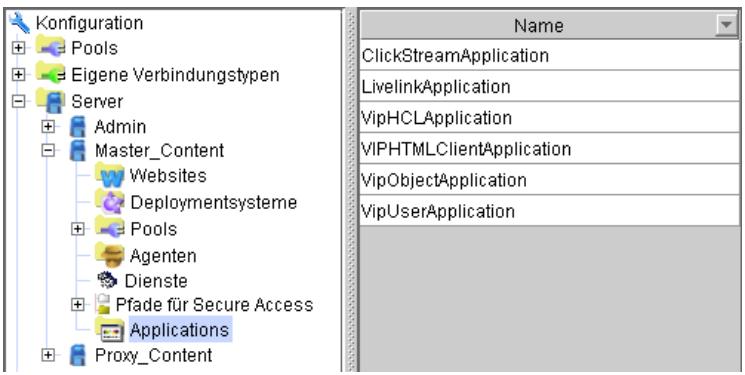


Abb. 5 – Applications des Content-Servers

Parameter der Application `WebServiceApplication` festlegen

Damit Benutzer über WCM WebServices auf die Inhalte Ihres WCM-Systems zugreifen können, müssen Sie den Zugriff auf die betreffenden Websites explizit erlauben. Dies erreichen Sie über entsprechende Parametereinstellungen der Application `WebServiceApplication`.

Die Parameter definieren Sie über *Konfiguration → Applications → WebServiceApplication → Parameter → {Parametername}*.

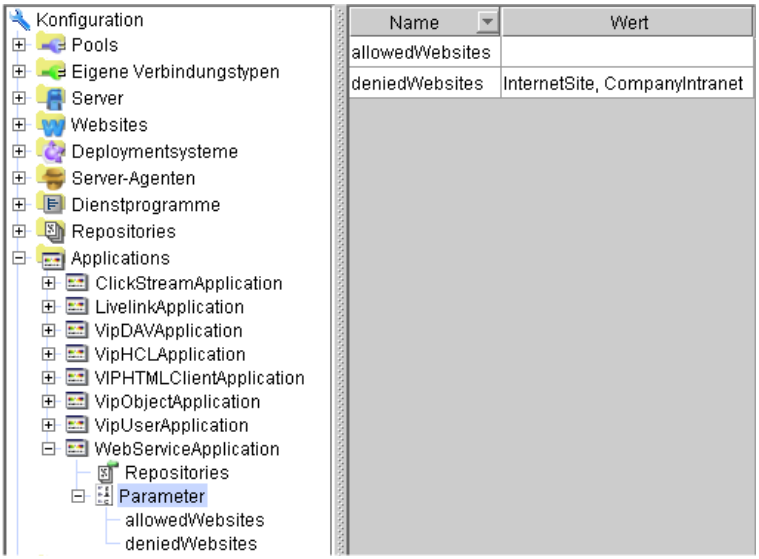


Abb. 6 – Parameter der Application `WebServiceApplication`

Die Application `WebServiceApplication` verfügt standardmäßig über die Parameter `allowedWebsites` und `deniedWebsites`.

- `allowedWebsites`: Geben Sie eine durch Komma separierte Liste von Website-Namen an. Auf diese Websites kann über WCM WebServices zugegriffen werden. Durch Eingabe eines Sternchens (*) erlauben Sie den Zugriff auf alle WCM-verwalteten

Websites. Der Standardwert ist leer, d.h., es ist kein Zugriff über WCM WebServices möglich.

- `deniedWebsites`: Geben Sie eine durch Komma separierte Liste von Website-Namen an. Auf diese Websites kann über WCM WebServices **nicht** zugegriffen werden. Durch Eingabe eines Sternchens (*) verbieten Sie den Zugriff auf alle WCM-verwalteten Websites. Ein Verbot hat Vorrang vor einer Erlaubnis. Der Standardwert ist leer.

3.2 Die WCM WebServices-Beschreibung

Die Beschreibung von WCM WebServices wird in Form einer WSDL-Datei geliefert. Dieser Abschnitt beschreibt, welche Informationen die Datei enthält und wie Sie auf diese zugreifen.

Auf die WCM WebServices-Beschreibung zugreifen

Die WSDL-Datei, die die WCM WebServices-Beschreibung enthält, wird dynamisch erzeugt. Um auf die Datei zuzugreifen, geben Sie folgende URL in Ihren Browser bzw. Ihr verwendetes Werkzeug ein:

```
{Basis-URL des HTTP-Servers}/{Verzeichnis der Webanwendung}/
      WebService/Port/{Website-Name}/{Datenhaltungssicht}/
      {Name des Deploymentsystems}?WSDL
```

Beispiel

http://wcmserver.company.example/wcm/WebService/Port/InternetSite/edit/InternetSite_edit?WSDL

Erläuterung zu einzelnen Bestandteilen der URL

- `?WSDL`: Dieser Parameter in der URL bewirkt die Generierung der WSDL-Datei, die zum Benutzer bzw. Dienstnehmer übertragen wird.
- `{Datenhaltungssicht}`: Der Zugriff auf eine Website über WCM WebServices bezieht sich immer auf eine bestimmte Datenhaltungssicht. Mögliche Werte sind:
  - Edit-Sicht: `edit`
  - QS-Sicht: `qa`
  - Produktionssicht: `prod`

  Die verschiedenen Sichten auf die Website-Daten werden von den entsprechenden Deploymentsystemen erzeugt.

  Für die Generierung der WSDL-Datei sind die Informationen zur Datenhaltungssicht und zum Namen des Deploymentsystems nicht zwingend erforderlich. Geben Sie die Sicht auf die Website-Daten nicht an, werden standardmäßig die Daten zur Edit-Sicht übertragen.
- `{Name des Deploymentsystems}`: Geben Sie den Namen des Deploymentsystems an, auf dessen Daten Sie zugreifen möchten. Das Deploymentsystem muss auf einem Content-Server installiert sein, der im Kontext einer JSP-Engine bzw. als Webanwendung in einem Application-Server läuft.

  Wenn Sie kein Deploymentsystem angeben, wird dies automatisch bestimmt. Dabei sucht der Content-Server nach einem Deploymentsystem, dessen Basis-URL der des angebotenen Web-Service ähnelt. Auf diese Weise soll auch bei fehlender Angabe des Deploymentsystems gewährleistet werden, dass der Web-Service-Client URLs übermittelt bekommt, mit denen er auf die generierten Seiten zugreifen kann.

Hinweis: Um den Zugriff auf die Seiten genau zu steuern, sollten Sie den exakten Namen des Deploymentsystems mit der gewünschten Datenhaltungssicht explizit in der URL angeben.

Inhalt der WCM WebServices-Beschreibung

Die WSDL-Datei zur Beschreibung von WCM WebServices enthält folgende Informationen:

- Liste der verfügbaren Methoden mit den erforderlichen Eingangsparametern und den Rückgabewerten
- WCM-spezifische Datentypen
- URL, über die der Web-Service-Proxy auf den Content-Server zugreift

Die Funktions- bzw. Methodenaufrufe erfolgen ausschließlich über entfernte Prozeduraufrufe. Die Eingangsparameter jedes Aufrufs enthalten die Daten, die der Content-Server für die jeweiligen Methodenaufrufe benötigt. Anschließend liefert der Web-Service-Proxy sämtliche Daten über den Rückgabewert der aufgerufenen Methode an den Client zurück.

Die WCM WebServices-Beschreibung liefert außerdem sämtliche WCM-spezifischen Datentypen, die in den Eingangsparametern oder im Rückgabewert enthalten sein können. Die wichtigsten Datentypen werden im Abschnitt 5.1 “WCM-spezifische Datentypen” ab Seite 76 näher beschrieben.

Mit folgender URL greift der Web-Service-Proxy auf den Content-Server zu:

```
{Basis-URL des HTTP-Servers}/{Verzeichnis der Webanwendung}/
      WebService/Port/{Website-Name}/{Datenhaltungssicht}/
      {Name des Deploymentsystems}
```

3.3 Fehlerbehandlung

Für das Auswerten von Fehlersituationen sind folgende Elemente wichtig:

- SOAP-Fehlermeldungen
- Fehlermeldungen (*Exceptions*) der WCM-Server

Aufbau von SOAP-Fehlermeldungen

Falls der WCM-Server einen Methodenaufruf nicht erfolgreich ausführen kann, wird eine SOAP-Fehlermeldung an den Dienstnehmer zurückgegeben. Diese SOAP-Fehlermeldung enthält ein `Fault`-Element, das wiederum folgende Elemente enthält:

- `faultcode`: Hier steht hinter der allgemeinen Bezeichnung `Server` der Typ der aufgetretenen Fehlersituation, z.B. `AccessDeniedException`. Die möglichen Exceptions des Content-Servers sind in Tabelle 2 “Exceptions und mögliche Ursachen” auf Seite 49 aufgelistet.
- `faultstring`: Beschreibung der Fehlersituation und eventuell der Ursachen als Klartext. Diese Meldung ist lokalisiert, d.h., sie wird in der Sprache verfasst, die im Admin-Client für den angemeldeten Benutzer eingestellt ist.

- `faultactor`: Dieses Element enthält die Bezeichnung “Vip-Content-Manager-API”, um dem Dienstnehmer zu zeigen, dass die Ausnahmesituation bei der Bearbeitung im WCM-Server und nicht beim Verbindungsaufbau, bei der Umwandlung der XML-Beschreibung der Eingabedaten oder der Transformation der Rückgabewerte in eine SOAP-Nachricht aufgetreten ist.
- `detail`: Dieses Element enthält eine Folge von lokalisierten Nachrichten, die in komplexen Fehlersituationen die Suche nach den Ursachen erleichtern.

Es folgt ein Beispiel für eine SOAP-Fehlermeldung, die dadurch entstanden ist, dass der angemeldete Benutzer “admin” nicht das Recht besitzt, über `release` ein WCM-Objekt freizugeben.

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:tns="http://gaussvip.com/" xmlns:types="http://gaussvip.com/
     encodedTypes" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<soap:Fault>
<faultcode>tns:Server.AccessDeniedException</faultcode>
<faultstring>0474: Der Benutzer &apos;admin&apos; besitzt nicht das
   Recht &apos;Freigeben&apos;.</faultstring>
<faultactor>Vip-Content-Manager-API</faultactor>
<detail>
<tns:message sequenceNo="0">0303: Der Master-Server konnte die Anfrage
    nicht ausführen.</tns:message>
<tns:message sequenceNo="1">0635: Das Freigeben von Objekt
    &apos;171&apos; ist fehlgeschlagen.</tns:message>
<tns:message sequenceNo="2">0300: Während der Vorbereitung der Aktion
    &apos;Freigeben&apos; auf Objekt &apos;171&apos; in der Website
      &apos;Susi&apos; trat ein Fehler auf.</tns:message>
<tns:message sequenceNo="3">0474: Der Benutzer &apos;admin&apos; besitzt
    nicht das Recht &apos;Freigeben&apos;.</tns:message>
</detail>
</soap:Fault>
</soap:Body>
</soap:Envelope>
```

Exceptions des Content-Servers

Die Fehlersituationen auf dem Content-Server werden durch folgende Exceptions klassifiziert:

Tabelle 2 – Exceptions und mögliche Ursachen

Name der Exception	Mögliche Ursache
`AccessDenied Exception`	Die aufgerufene Methode erfordert eine Aktion, für die der angemeldete Benutzer keine Berechtigung hat.
`ActionNotPermitted Exception`	Das Ausführen einer Methode wird verweigert, da dies zu einer Regelverletzung führen würde. Dies kann z. B. der Fall sein, wenn ein WCM-Objekt vorgelegt werden soll, dessen übergeordnetes Thema noch nicht in der QS-Sicht sichtbar ist. Eine andere Ursache könnte sein, dass das Ausführen der Methode zu Namenskonflikten führen würde.
`AttributeException`	Diese Exception tritt auf, wenn ein unbekanntes Attribut übergeben wird, ein Attribut geändert werden soll, das der Benutzer nicht ändern darf, oder für das Attribut ein von der Website-Konfiguration abweichender Datentyp verwendet wird.
`DatabaseException`	Beim Zugriff auf das von Livelink WCM Server genutzte RDBMS traten Probleme auf.
`FileException`	Beim lesenden oder schreibenden Zugriff auf eine Datei traten Probleme auf.
`InvalidContextId Exception`	WCM WebServices verwaltet für jeden angemeldeten Dienstnehmer eine Kontext-Kennung. Wenn diese ungültig wird, dann wird diese Exception gemeldet. Der Dienstnehmer muss sich erneut mit Benutzerkennung und Passwort anmelden.

Name der Exception	Mögliche Ursache
`InvalidObject Exception`	Werden die Attribute eines WCM-Objekts von zwei Benutzers nahezu gleichzeitig geändert, dann kann es passieren, dass der Benutzer, dessen Änderung zuletzt bearbeitet wird, seine Transaktion intern mit einem veralteten Datensatz beginnt. Dies führt zu dieser Exception. Im Fall von WCM WebServices ist dies jedoch unwahrscheinlich, da die internen Objektdaten erst kurz vor der Änderung vom Content-Server geholt werden.
`LicenseException`	Die Lizenzen für Livelink WCM Server sind abgelaufen.
`LoginException`	Der Dienstnehmer hat sich mit einer ungültigen Benutzerkennung oder einem falschen Passwort am WCM-System angemeldet.
`MailException`	Probleme beim Versenden von E-Mails im Rahmen des Stagings, z.B. weil ein Mail-Server nicht erreicht wurde.
`NetException`	Bei der Nutzung von Netzwerkverbindungen durch den WCM-Server, beim Verbindungsaufbau oder beim Datenaustausch traten Probleme auf. Verbindungsprobleme zwischen dem WCM WebServices-Dienstnehmer und der Applikation des Dienstanbieters sind nicht der Grund für diese Exception.
`ObjectInUseException`	Das WCM-Objekt, das geändert werden soll, wird gerade von einem anderen Benutzer des WCM-Systems bearbeitet.
`ObjectNotFound Exception`	Der angemeldete Benutzer hat kein Leserecht für ein Objekt, die beim Zugriff auf ein WCM-Objekt angegebene OID/Versionsnummer ist nicht gültig oder das entsprechende Objekt konnte nicht in der Website gefunden werden.

Name der Exception	Mögliche Ursache
`RunlevelException`	Der angesprochene WCM-Server oder die Website befinden sich in einem Runlevel, in dem die gewünschte Operation nicht möglich ist. Der Runlevel wird u. a. für Wartungsarbeiten am WCM-System geändert.
`UnexpectedException`	Bei der Ausführung einer Methode traten interne Fehler auf, die keinem der anderen Fehlertypen zugeordnet werden können. Siehe auch `VipApiException`.
`VetoException`	Ein auf dem WCM-Server laufender Agent hat die Ausführung der aufgerufenen Methode durch ein Veto verhindert. Agenten können z. B. die Ausführung bestimmter Staging-Aktionen an ausgewählten WCM-Objekten in einem bestimmten Zeitraum verhindern.
`VipApiException`	Kann ein Fehler keinem der anderen Fehlertypen zugeordnet werden, tritt diese Exception auf.
`WorkflowException`	Wurde im Workflow ein Übergang ausgewählt, der nicht ausgeführt werden kann, tritt diese Exception auf.

KAPITEL 4

Zugriff auf Administrationsdaten und Authentifizierung

Mithilfe von WCM WebServices können Sie auf die Daten und Funktionen des Administrationsservers des WCM-Systems zugreifen:

- Ermitteln der Benutzer-, Gruppen- und Rollenprofilen mit den entsprechenden Einstellungen und Zuordnungen (siehe Abschnitt 4.1 “Benutzer, Gruppen und Rollen” auf Seite 54)
- Ermitteln der Funktionsbereiche von Principals (siehe Abschnitt 4.2 “Funktionsbereiche” auf Seite 62)
- Ermitteln von Informationen über Websites (siehe Abschnitt 4.3 “Websites” auf Seite 65)
- Ermitteln von Informationen über Deploymentsysteme (siehe Abschnitt 4.4 “Deploymentsysteme” auf Seite 66)
- Auslesen des Systemzustands (Runlevel) von WCM-Servern und Websites (siehe Abschnitt 4.5 “Runlevels” auf Seite 67)
- Ermitteln anderer Informationen des WCM-Systems wie Version und Sprachen (siehe Abschnitt 4.6 “Allgemeine Abfragen” auf Seite 70)

Hinweis: Der Zugriff auf die Daten des Administrationsservers setzt Administrationsrechte voraus, z. B. das Recht “Zugriff auf Benutzerverwaltung” für das Lesen von Benutzer-, Gruppen- und Rollenprofilen und “Zugriff auf Systemverwaltung” für das Lesen von Systeminformationen. Weitere Informationen zu den Administrationsrechten finden Sie im Livelink WCM Server-Administratorhandbuch.

Um die Funktionen von Livelink WCM Server über WCM WebServices nutzen zu können, ist außerdem die Anmeldung am WCM-System erforderlich. Lesen Sie dazu Abschnitt 4.7 “Authentifizierung” auf Seite 71.

4.1 Benutzer, Gruppen und Rollen

Sie können über WCM WebServices die Einstellungen (Profile) von Benutzern, Gruppen und Rollen auslesen und die Standard-Objektrechte von Principals setzen.

Datentypen für Principals

Für die Übergabe der Profile von Principals werden die Datentypen `User`, `Group` und `Role` benutzt. Alle diese Datentypen basieren auf dem Datentyp `Principal`.

Die folgende Tabelle zeigt die Komponenten des Datentyps `User`.

Tabelle 3 – Datentypen für Benutzer

Attribut	Datentyp	Beschreibung
`name`	`String`	Benutzerkennung
`hasProfile`	`boolean`	Zeigt an, ob weitere Profilinformation enthalten ist
`commonName`	`String`	Kompletter Name des Benutzers
`eMailAddress`	`String`	E-Mail-Adresse des Benutzers
`locale`	`Locale`	Sprache des Benutzers
`websites`	`String[]`	Liste der Website-Namen, denen der Benutzer zugeordnet ist
`subsitute`	`User`	Stellvertreter, der das Recht hat, sich unter diesem Benutzer anzumelden
`substituteOf`	`User[]`	Menge von Benutzern, für die dieser Benutzer der Stellvertreter ist
`groups`	`Group[]`	Liste der Gruppen, zu denen der Benutzer gehört
`roles`	`Role[]`	Liste der Rollen, zu denen der Benutzer gehört
`initially Granted Permissions`	`Permission[]`	Erlaubte Standard-Objektrechte des Benutzers. Diese Rechte werden als Vorgabe eingetragen, wenn der Benutzer der Zugriffssteuerungsliste (ACL) eines WCM-Objekts hinzugefügt wird.

Attribut	Datentyp	Beschreibung
initially Denied Permissions	Permission[]	Verbotene Standard-Objektrechte des Benutzers. Diese Rechte werden als Vorgabe eingetragen, wenn der Benutzer der Zugriffs-steuerungsliste (ACL) eines WCM-Objekts hinzugefügt wird.
trustedLogin	boolean	Zeigt an, ob für diesen Benutzer die vertraute Anmeldung erlaubt ist. Diese Einstellung macht es möglich, dass sich ein Benutzer nach einer Anmeldung für andere Produkte von Livelink WCM Server nicht noch einmal authentifizieren muss (single sign-on).
vipAccess	boolean	Zeigt an, ob dem Benutzer der Zugang zum WCM-System gestattet ist

Die Datentypen Group und Role sind identisch aufgebaut. Ihre Komponenten sind in der folgenden Tabelle aufgeführt.

Tabelle 4 – Datentypen für Gruppen und Rollen

Attribut	Datentyp	Beschreibung
name	String	Eindeutiger Name der Gruppe bzw. Rolle
hasProfile	boolean	Zeigt an, ob weitere Profilinformation enthalten ist
eMailAddress	String	E-Mail-Adresse der Gruppe bzw. Rolle
websites	String[]	Liste der Website-Namen, denen die Gruppe bzw. Rolle zugeordnet ist

Attribut	Datentyp	Beschreibung
`members`	`User[]`	Liste der Benutzer, die zu dieser Gruppe bzw. Rolle gehören
`initially Granted Permissions`	`Permission[]`	Erlaubte Standard-Objektrechte der Gruppe bzw. Rolle. Diese Rechte werden als erlaubt eingetragen, wenn die Gruppe bzw. Rolle der Zugriffs-steuerungsliste (ACL) eines WCM-Objekts hinzugefügt wird.
`initially Denied Permissions`	`Permission[]`	Verbotene Standard-Objektrechte der Gruppe bzw. Rolle. Diese Rechte werden als verboten eingetragen, wenn die Gruppe bzw. Rolle der Zugriffssteuerungsliste (ACL) eines WCM-Objekts hinzugefügt wird.

Methoden zum Auslesen von Profilen

Das Profil eines Benutzers, einer Gruppe oder einer Rolle wird über die Methoden `getUserProfiles`, `getGroupProfiles` und `getRoleProfiles` ermittelt. Diese Methoden liefern jeweils ein Array aus dem Datentyp `User`, `Group` bzw. `Role` zurück. Dieses Array enthält genau die Principals, die den übergebenen Filterbedingungen entsprechen.

Zur Festlegung der Filterbedingung werden Daten vom Typ `Filter` und den davon abgeleiteten Datentypen benutzt. Diese Datentypen werden im Abschnitt 5.6 “Suchen nach WCM-Objekten” ab Seite 147 beschrieben.

Für die Suche nach Benutzern, Gruppen oder Rollen gelten die folgenden Einschränkungen:

- Es können nur `EqualFilter`, `LikeFilter` und `StringContainsFilter` verwendet werden. Für die logische Verknüpfung sind `AndFilter`, `OrFilter` und `NotFilter` zugelassen.

- Es können Bedingungen für die Attribute `name`, `commonName`, `eMailAddress`, `locale`, `trustedLogin` und `vipAccess` angegeben werden.

`getGroupProfiles`

Liefert die Daten der Gruppen zurück, die auf das spezifizierte Filterkriterium zutreffen. Die Ergebnismenge kann durch `startResult` und `numberOfResults` begrenzt werden.

Parameter

Tabelle 5 – Parameter der Methode `getGroupProfile`

Parameter	Datentyp	Beschreibung
`ldapContext`	`String`	LDAP-Kontext, in dem die Suche durchgeführt wird, oder `null`, falls die Benutzerverwaltung nicht LDAP-basiert ist
`sortList`	`Sort[]`	Liste der Attribute, nach denen die Ergebnisliste sortiert werden soll (in nach Priorität absteigender Folge)
`filter`	`Filter`	Suchkriterium oder `null`, um nach allen Gruppen zu suchen
`startResult`	`int`	Zahl, die das erste Element des Suchergebnisses angibt Mit 0 wird das Suchergebnis von Anfang an geliefert.
`numberOfResults`	`int`	Anzahl der zurückzuliefernden Suchergebnisse beginnend bei `startResult` Mit `-1` werden alle Ergebnisse zurückgeliefert.

Rückgabe

`Group[] groups`: Liste von Gruppen, die dem angegebenen Suchkriterium entsprechen

getRoleProfiles

Liefert die Daten der Rollen zurück, die auf das spezifizierte Filterkriterium zutreffen. Die Ergebnismenge kann durch `startResult` und `numberOfResults` begrenzt werden.

Parameter

Tabelle 6 – Parameter der Methode `getRoleProfile`

Parameter	Datentyp	Beschreibung
`ldapContext`	`String`	LDAP-Kontext, in dem die Suche durchgeführt wird, oder `null`, falls die Benutzerverwaltung nicht LDAP-basiert ist
`filter`	`Filter`	Suchkriterium oder `null`, um nach allen Rollen zu suchen
`sortList`	`Sort[]`	Liste der Attribute, nach denen die Ergebnisliste sortiert werden soll (in nach Priorität absteigender Folge)
`startResult`	`int`	Zahl, die das erste Element des Suchergebnisses angibt Mit 0 wird das Suchergebnis von Anfang an geliefert.
`numberOfResults`	`int`	Anzahl der zurückzuliefernden Suchergebnisse beginnend bei `startResult` Mit `-1` werden alle Ergebnisse zurückgeliefert.

Rückgabe

`Role[] roles`: Liste von Rollen, die auf das angegebene Suchkriterium passen

getUserProfiles

Liefert die Benutzerdaten der Benutzer zurück, die auf das spezifizierte Filterkriterium zutreffen. Die Ergebnismenge kann durch `startResult` und `numberOfResults` begrenzt werden.

Parameter

Tabelle 7 – Parameter der Methode `getUserProfile`

Parameter	Datentyp	Beschreibung
`ldapContext`	`String`	LDAP-Kontext, in dem die Suche durchgeführt wird, oder `null`, falls die Benutzerverwaltung nicht LDAP-basiert ist
`filter`	`Filter`	Suchkriterium oder `null`, um nach allen Benutzern zu suchen
`sortList`	`Sort[]`	Liste der Attribute, nach denen die Ergebnisliste sortiert werden soll (in nach Priorität absteigender Folge)
`startResult`	`int`	Zahl, die das erste Element des Suchergebnisses angibt Mit 0 wird das Suchergebnis von Anfang an geliefert.
`numberOfResults`	`int`	Anzahl der zurückzuliefernden Suchergebnisse beginnend bei `startResult` Mit `-1` werden alle Ergebnisse zurückgeliefert.

Rückgabe

`User[] users`: Liste von Benutzern, die auf das angegebene Suchkriterium passen

Standard-Objektrechte übergeben

Die Komponente `initialGrantedPermissions` der Datentypen `User`, `Group` und `Role` enthält eine Liste mit erlaubten Standard-Objektrechten für den Principal. Diese voreingestellten Zugriffsrechte sollten von einem WCM WebServices-Dienstprogramm als Einstellung vorgeschlagen werden, wenn der Principal in die Zugriffssteuerungsliste eines WCM-Objekts aufgenommen wird. Die Komponente `InitialDeniedPermissions` wird analog verwendet und beinhaltet die verbotenen Standard-Objektrechte des Principals.

Die Änderung der Zugriffssteuerungsliste eines WCM-Objekts erfolgt mithilfe des Datentyps `Acl`, siehe “ACL eines WCM-Objekts ändern” auf Seite 91.

4.2 Funktionsbereiche

In Livelink WCM Server müssen Benutzer direkt oder indirekt (über die Gruppen- bzw. Rollenzugehörigkeit) Funktionsbereichen zugeordnet sein, um bestimmte Aktionen im Rahmen der Website-Verwaltung durchzuführen. Allgemeine Informationen zu Funktionsbereichen erhalten Sie im Abschnitt “Funktionsbereiche” auf Seite 36.

In WCM WebServices werden Funktionsbereiche durch den Datentyp `FunctionalArea` repräsentiert. Die Methode `getFunctionalAreas` liefert alle innerhalb des WCM-System definierten Funktionsbereiche.

Der Datentyp `FunctionalArea` enthält den Namen des Funktionsbereichs und die Mengen von Benutzern, Gruppen und Rollen, die diesem Funktionsbereich **direkt** zugeordnet sind.

Die folgende Tabelle zeigt die verfügbaren Standard-Funktionsbereiche und die dazugehörigen Namen. Über den Admin-Client können eigene Funktionsbereiche angelegt und Objekttypen zugeordnet werden. Diese Funktionsbereiche werden durch ihren Namen repräsentiert.

Tabelle 8 – Namen der Standard-Funktionsbereiche

Funktionsbereich	Konstante	Erlaubte Ansicht oder Funktion im Content-Client
Basis	`VIP`	Anlegen, Ausleihen und Zurückgeben von Objekten, die auf den zugeordneten Objekttypen basieren
Fortgeschritten	`ADVANCED`	
Dynamisch	`DYNAMIC`	
Formular	`FORM`	
Workflow	`WORKFLOW`	Zuordnung von Workflows zu Objekten herstellen und aufheben

Funktionsbereich	Konstante	Erlaubte Ansicht oder Funktion im Content-Client
Intelligente Vorlagen	ITF	Wird nicht standardmäßig verwendet, dient der Abwärtskompatibilität zu VIP 5e
Direkte Freigabe	DIRECT_RELEASE	Bearbeiten der Option *Direkte Freigabe* in den Metadaten
Dialog Referenzen	REFERENCES	Ansicht des Referenzen-Dialogs
Dialog Zugriffsrechte	ACCESS_RIGHTS	Ansicht des Zugriffsrechte-Dialogs
Dialog Protokoll	LOG	Ansicht des Protokoll-Dialogs
Livelink	LIVELINK	Ansicht des Livelink-Metadaten-Dialogs
Filter Standard	FILTER_STANDARD	Verwenden der Standardfilter
Filter Bearbeiten	FILTER_EDIT	Erstellen und Bearbeiten von Filtern im Filtereditor
Ansicht Untergeordnete Objekte	OBJECTLIST	Ansicht “Untergeordnete Objekte”
Ansicht Objektliste	LISTVIEW	Ansicht “Objektliste”
Ansicht Meine Objekte	FILTER_TODO	Ansicht “Meine Objekte” und Ansicht “Mein Eingangskorb”

Funktionsbereich	Konstante	Erlaubte Ansicht oder Funktion im Content-Client
Ansicht Vorlagenstruktur	`TEMPLATE_STRUCTURE`	Ansicht “Vorlagenstruktur”
Importieren	`IMPORT`	Verwenden der Import-funktionen
Suche	`COMI_SEARCH`	Verwenden der Such-funktionen

Methoden zum Auslesen von Funktionsbereichen

`getFunctionalAreas`

Liefert alle im WCM-System verfügbaren Funktionsbereiche zurück

Parameter

keine

Rückgabe

`FunctionalArea[] functionalAreas`: die verfügbaren Funktionsbereiche

4.3 Websites

Eine Website innerhalb eines WCM-Systems wird eindeutig durch einen Namen identifiziert. WCM WebServices bietet die Methode `getWebSiteNames`, um die Namen aller Websites auszulesen, die im WCM-System verwaltet werden. Jeder Website sind im Allgemeinen mehrere Deploymentsystem zugeordnet.

Methoden für Websites

`getAttributeSets`

Liefert alle der Website zugeordneten Attributmengen zurück

Parameter

keine

Rückgabe

`AttributeSet[] attributeSets`: die verfügbaren Attributmengen

`getObjectCategories`

Liefert alle der Website zugeordneten Objektkategorien zurück

Parameter

keine

Rückgabe

`ObjectCategory[] objectCategories`: die verfügbaren Objektkategorien

`getObjectTypes`

Liefert alle der Website zugeordneten Objekttypen zurück

Parameter

keine

Rückgabe

`ObjectType[] objectTypes`: die verfügbaren Objekttypen

`getWebsiteNames`

Liefert die Namen aller im WCM-System verfügbaren Websites zurück

Parameter

keine

Rückgabe

`string[] websiteNames`: die Namen der verfügbaren Websites

4.4 Deploymentsysteme

Es gibt drei verschiedene Typen von Deploymentsystemen entsprechend den Datenhaltungssichten Edit, QS und Produktion. Ein Deploymentsystem für die Edit-Sicht verfügt über alle Ausprägungen der WCM-Objekte und verwendet diese für die Seitengenerierung. Im Qualitätssicherungs- bzw. Produktionsdeploymentsystem sind nur die entsprechenden Daten der jeweiligen Sicht vorhanden.

Die wesentlichen Eigenschaften eines Deploymentsystems sind wie folgt:

- die Website, auf der das Deploymentsystem basiert
- der Server, auf dem das Deploymentsystem ausgeführt wird
- das Verzeichnis auf dem o.g. Server, in dem die generierten Seiten gespeichert werden sollen
- die Basis-URL, die für die Generierung von Verweisen verwendet werden soll

Mit der Methode `getDeploymentSystems` kann die Liste aller Deploymentsysteme für eine Website ermittelt werden. Pro Deploymentsystem werden der Name und der Typ an den Dienstnehmer zurückgegeben.

`getDeploymentSystems`

Liefert alle Deploymentsysteme für eine angegebene Website zurück

Parameter

Tabelle 9 – Parameter der Methode `getDeploymentSystems`

Parameter	Datentyp	Beschreibung
`website`	`String`	Name der Website

Rückgabe

`DeploymentSystem[] deploymentSystems`: Liste der verfügbaren Deploymentsysteme

4.5 Runlevels

Der Runlevel bezeichnet den aktuellen Zustand eines Servers oder einer Website, der definiert, welche funktionalen Komponenten des Servers oder der Website gerade aktiviert bzw. deaktiviert sind. Die Runlevels sind unterteilt in Server-Runlevels und Website-Runlevels, wobei die Website-Runlevels auf den Server-Runlevels aufbauen. Weitere Angaben zu den Runlevels finden Sie im Livelink WCM Server-Administratorhandbuch.

Die Runlevels können mit der Methode `listAllRunlevels` ausgelesen werden. Beim Start eines Servers werden die Runlevels von Runlevel 0 bis Runlevel 10 durchlaufen, beim Herunterfahren des Servers entsprechend von 10 bis 0. Darüberhinaus können die Runlevels von Servern und Websites über den Admin-Client von Livelink WCM Server separat gesteuert werden.

Mit der Methode `getRunlevel` kann der aktuelle Runlevel der Website, die dem Web-Service zugeordnet ist, ermittelt werden (d.h. die Runlevels ab Runlevel 6).

Folgende Runlevels stehen für Server zur Verfügung:

- **0 Server nicht erreichbar:** Der Server ist nicht erreichbar.
- **1 Verbindungen geschlossen:** Alle Kommunikationsverbindungen sind geschlossen. Datenbank- und LDAP-Verbindungen sowie Dienste sind nicht mehr verfügbar. Beim Übergang zum nächsthöheren Runlevel wird die Verbindungsverwaltung aufgebaut.
- **2 Keine Benutzer angemeldet:** Es sind keine Benutzer mehr an diesem Server angemeldet. Beim Übergang zum nächsthöheren Runlevel wird die Benutzerverwaltung aktiviert.
- **3 Keine Agenten aktiv:** Alle Server-Agenten sind beendet. Beim Übergang zum nächsthöheren Runlevel werden alle Server-Agenten gestartet.
- **4 Einzelnutzer-Betrieb:** Alle Benutzer bis auf den Administrator, der die Runlevel-Änderung vornimmt, werden abgemeldet und können sich nicht mehr am WCM-System anmelden. Dieser Runlevel ist besonders für Wartungsarbeiten am WCM-System vorgesehen. Beim Übergang zum Runlevel 3 werden die Websites vollständig heruntergefahren. Beim Übergang zum nächsthöheren Runlevel wird das System für alle Benutzer freigegeben.
- **5 Server läuft:** Der Server ist vollständig gestartet.

Folgende Runlevels von Websites gibt es. Für die Website-Runlevel wird mindestens der Server-Runlevel 4 vorausgesetzt.

- **6 Website nicht zugreifbar:** Eine bzw. alle Websites sind nicht mehr verfügbar (auch nicht für lesende Zugriffe). Damit sind auch die Deploymentsysteme nicht mehr aktiv.

- **7 Website wird konfiguriert:** Die angegebene Website wird in diesem Runlevel konfiguriert (die Konfigurationsdaten werden geschrieben). Beim Übergang zum nächsthöheren Runlevel werden die Websites für den lesenden Zugriff initialisiert.
- **8 Website schreibgeschützt:** Auf eine bzw. auf alle Websites kann lesend zugegriffen werden. Vom Deploymentsystem abhängige Daten wie z.B. URLs und Pfade sind in diesem Runlevel noch nicht verfügbar. Beim Übergang zum nächsthöheren Runlevel werden die Deploymentsysteme initialisiert.
- **9 Deployment abgeschlossen:** Die Deploymentsysteme sind vollständig initialisiert, d.h. Deploymentaufgaben werden durchgeführt. Beim Übergang zu Runlevel 8 werden keine Deploymentaufgaben mehr durchgeführt. Beim Übergang zum nächsthöheren Runlevel werden die Websites für den schreibenden Zugriff freigegeben. Die Verteilung von WCM-Objekten an Proxy-Content-Server findet erst im nächsten Runlevel statt.
- **10 Website läuft:** Eine bzw. alle Websites stehen für den schreibenden Zugriff zur Verfügung.

Methoden zum Auslesen von Runlevels

`getRunlevel`

Liefert den Runlevel der Website zurück, die dem Web-Service zugeordnet ist

Parameter

keine

Rückgabe

`Runlevel runlevel`: der Runlevel der Website

`listAllRunlevels`

Liefert alle in einem WCM-System möglichen Runlevels zurück

Parameter

keine

Rückgabe

`Runlevel[] runlevels`: die möglichen Runlevels

4.6 Allgemeine Abfragen

Für allgemeine Abfragen stehen folgende Methoden zur Verfügung:

`getLanguages`

Liefert alle vom Server unterstützten Sprachen zurück

Parameter

keine

Rückgabe

`Locale[] languages`: Liste der verfügbaren Sprachen

`getVIPVersion`

Liefert die Version von Livelink WCM Server zurück

Parameter

keine

Rückgabe

`KernelVersion version`: Version von Livelink WCM Server

4.7 Authentifizierung

Der Zugriff auf ein WCM-System über WCM WebServices erfordert eine Benutzerauthentifizierung, um die einzelnen Methoden überhaupt nutzen zu können. Bei vielen Aktionen werden die Funktionsbereiche des Benutzers bzw. die Zugriffssteuerungslisten der WCM-Objekte geprüft, um festzustellen, ob der Benutzer über die notwendigen Rechte verfügt.

Die Authentifizierung erfolgt bei der Anmeldung des Web-Service am WCM-Server. Sie führt im WCM-System zur Erzeugung eines bei jeder Anmeldung unterschiedlichen, innerhalb eines WCM-Systems eindeutigen `ContextId`-Objekts. Damit ist der Web-Service eindeutig im System registriert. Die `ContextId` bleibt im WCM-System so lange erhalten, wie die zum aktuellen Web-Service gehörige Session existiert.

Es ist für WCM WebServices keine weitere `login`-Methode erforderlich. Die Übergabe von Benutzername und Passwort für die Anmeldung am WCM-System erfolgt über die auf Grundlage von WCM WebServices programmierte Anwendung. Name und Passwort müssen dazu im Header des HTTP-Requests eingetragen werden.

Für das Beenden einer Session, die Authentifizierung als Stellvertreter und das Ändern von Passwörtern stehen folgende Methoden zur Verfügung:

`changePassword`

Ändert das Passwort für den angegebenen Benutzer

Parameter

Tabelle 10 – Parameter der Methode `changePassword`

Parameter	Datentyp	Beschreibung
`newPassword`	`String`	Neues Passwort

Rückgabe

keine

logout

Meldet den Benutzer – bekannt durch seine `ContextId` – ab

Parameter

keine

Rückgabe

keine

substituteLogin

Meldet den bereits angemeldeten Benutzer – bekannt durch seine `ContextId` – als Vertreter des angegebenen Benutzers `userName` am WCM-System an. Der durch die `ContextId` repräsentierte Benutzer muss als Vertreter des angegebenen Benutzers arbeiten dürfen.

Parameter

Tabelle 11 – Parameter der Methode `substituteLogin`

Parameter	Datentyp	Beschreibung
`userName`	`String`	Kennung des Benutzers, in dessen Vertretung gearbeitet werden soll

Rückgabe

keine

KAPITEL 5
Objektverwaltung

In diesem Kapitel werden zunächst die wichtigsten Datentypen beschrieben, die für die Bearbeitung von WCM-Objekten über WCM WebServices genutzt werden. Anschließend werden die Methoden der Objektverwaltung vorgestellt. Dabei wird zwischen folgenden Hauptgruppen unterschieden:

- allgemeine Parameter wie `DeploymentWaitInfo` und `EMailInfo`, siehe Abschnitt 5.2 “Allgemeine Parameter” auf Seite 98
- Methoden für die Durchführung von Staging-Aktionen wie “Ausleihen” oder “Vorlegen”, siehe Abschnitt 5.3 “Staging-Methoden” auf Seite 100
- Methoden für die Duchführung von Content Workflow, siehe Abschnitt 5.4 “Workflow-Methoden” auf Seite 109
- Methoden für die Bearbeitung von WCM-Objekten z. B. zum Ändern der Metadaten, siehe Abschnitt 5.5 “Methoden der Objektverwaltung” auf Seite 119

> **Hinweis:** Da die Syntax des Methodenaufrufs von der verwendeten Programmiersprache abhängt, wurde eine allgemeine Form der Beschreibung von Methodennamen und Parametern gewählt.

Im letzten Abschnitt wird die Suche nach WCM-Objekten mithilfe von Filtern beschrieben, siehe Abschnitt 5.6 “Suchen nach WCM-Objekten” auf Seite 147.

5.1 WCM-spezifische Datentypen

Jedes WCM-Objekt wird durch eine Reihe von Daten repräsentiert. Um über WCM WebServices auf das WCM-System zuzugreifen, benötigen Sie Kenntnisse über die verwendeten Datentypen. Ein Datentyp legt dabei die möglichen Werte und deren Verwendung fest.

In Livelink WCM Server werden die Daten intern – sofern es sich nicht um Java-Basistypen wie Integer- oder Boolsche Werte handelt – durch Java-Klassen bzw. Java-Interface-Definitionen charakterisiert. In WCM WebServices sind diesen Datentypen XML-Schema-Datentypen zugeordnet. Zum Teil können dabei die Standard-XML-Schema “Simple Types” verwendet werden. Für andere WCM-Datentypen müssen WCM-spezifische “Complex Types” verwendet werden, die zum Namensraum “http://gaussvip.com/” gehören.

Eine vollständige Übersicht über die verwendeten Datentypen bietet die WSDL-Datei von WCM WebServices, siehe “Inhalt der WCM WebServices-Beschreibung” auf Seite 46. Im Folgenden werden die wichtigsten Datentypen beschrieben:

- OID und Versionen von WCM-Objekten, siehe “Datentyp: `ObjectId`” auf Seite 77
- Metadaten, siehe “Datentyp: `ObjectData`” auf Seite 78
- Objekttyp, siehe “Datentyp: `ObjectType`” auf Seite 84
- Objektstatus, siehe “Datentyp: `ObjectState`” auf Seite 86
- Objekt-Zugriffsrechte, siehe “Datentyp: `Permission`” auf Seite 88
- Zugriffsteuerungsliste, siehe “Datentyp: `Acl`” auf Seite 90
- Eintrag in der Zugriffssteuerungsliste, siehe “Datentyp: `AclEntry`” auf Seite 91

- Repräsentation von Baumstrukturen für Objekte, siehe “Datentyp: `MultiImportPart`” auf Seite 92
- Workflow-Aktionen, siehe “Datentypen für Workflow-Aktionen” auf Seite 94

Datentyp: `ObjectId`

Die Objekt-ID, die so genannte OID, repräsentiert eine Referenz auf ein WCM-Objekt. Sie ist ein eindeutiger Bezeichner für ein WCM-Objekt innerhalb einer Website. Die Eindeutigkeit bezieht sich nur auf die aktuelle Version eines WCM-Objekts. Archivierte Objekte haben dieselbe OID mit einer unterschiedlichen Versionsnummer.

Tabelle 12 – Die Komponenten des Datentyps `ObjectId`

Attribut	Datentyp	Beschreibung
`id`	`String`	Interne Repräsentation als Zeichenkette

Wenn ein WCM-Objekt eine Staging-Stufe wie Bearbeitung oder Qualitätssicherung durchläuft, entsteht eine neue Version des Objekts. Die Versionierung von WCM-Objekten erfolgt mithilfe des Datentyps `Version`. Eine Version besteht aus drei Nummern für Hauptversion (`major`), Nebenversion (`minor`) und Mikroversion (`micro`). Die Mikroversion ändert sich, wenn ein WCM-Objekt in der Edit-Sicht geändert wird (Ausleihen, Metadaten ändern). Die Nebenversion ändert sich, wenn ein WCM-Objekt zur Qualitätssicherung vorgelegt wird, wobei dann die Mikroversion auf 0 gesetzt wird. Die Hauptversion wird beim Freigeben erhöht, wobei die Nebenversion auf 0 gesetzt wird. Mithilfe der Methode `getVersionList` können Sie die verfügbaren Versionen eines WCM-Objekts bestimmen, siehe “getVersionList” auf Seite 140.

Datentyp: `ObjectData`

Allgemeine Informationen zu Objektdaten erhalten Sie im Abschnitt “Objektdaten” auf Seite 29.

Der Datentyp `ObjectData` enthält die Metadaten eines WCM-Objekts. Bei der Suche mit der `filter`-Methode wird ein `ObjectData`-Array zurückgegeben, das je ein Element pro gefundenes WCM-Objekt enthält. Diese Elemente enthalten dann genau die Attribute, die in dem Eingabeparameter `attributeKeys` der `filter`-Methode angegeben wurden.

Mit der `get`-Methode können Sie sich diese Daten genau für ein WCM-Objekt einer bestimmten Version vom Server holen, auch hier müssen Sie die gewünschten Attribute in der Liste `attributeKeys` vorher wählen.

Umgekehrt wird bei der `change`- und der `create`-Methode ein Parameter vom Typ `ObjectData` an den Server übergeben, um die Attribute zu ändern bzw. die Attribute eines neuen Objekts zu setzen.

Die Symbole in der folgenden Tabelle verdeutlichen die möglichen Aktionen für das entsprechende Metadatum:

Der Wert `null` ist für dieses Attribut erlaubt.

Nach dem Attribut kann gesucht werden, d.h., es kann für die Konstruktion von Filtern verwendet werden.

Der Attributwert kann geändert werden.

Tabelle 13 – Die Komponenten des Datentyps `ObjectData`

Attribut	Datentyp	Beschreibung
`ACL`	`Acl`	Zugriffssteuerungsliste des WCM-Objekts
`attributeKeys`	`String[]`	Werden die Objektdaten mit einer `create`- oder `get`-Methode zum Server gesendet, so muss dieses Array die Menge von Attributnamen enthalten, deren Werte vom Server gesetzt oder geändert werden sollen. Werden die Objektdaten vom Server zurückgeschickt, dann enthält dieses Array die Menge der Attributnamen, die tatsächlich vom Server gesetzt wurden. Alle anderen Attribute haben nicht-verlässliche Werte.
`children`	`ObjectId[]`	OIDs der untergeordneten Objekte (Kinder) des WCM-Objekts, falls das WCM-Objekt ein Thema ist. Falls das WCM-Objekt kein Thema ist oder es keine untergeordneten Objekte hat, ist dieser Wert `null`.
`compatible ObjectTypes`	`ObjectType[]`	Objekttypen, in die der Objekttyp des WCM-Objekts überführt werden kann
`contentSize`	`int`	Größe des Objektinhalts in Byte `-1` bedeutet, dass das Objekt keinen Inhalt hat

Attribut	Datentyp	Beschreibung
createdBy	User	Benutzer, der das WCM-Objekt angelegt hat
createdDate	dateTime	Datum (mit Uhrzeit), an dem das WCM-Objekt erstellt wurde
deploymentHint	String	Deploymenthinweis, d.h. der vorgeschlagene Name für die Seite, die aus dem WCM-Objekt erzeugt wird
description	String	Beschreibung des Objektinhalts
directRelease	boolean	Dieser Wert ist genau dann wahr, wenn das WCM-Objekt direkt freigegeben werden kann, ohne die Qualitätssicherung zu durchlaufen.
editEMail Receivers	String[]	E-Mail-Adressen, die zur Benachrichtigung verwendet werden sollen, wenn das WCM-Objekt abgelehnt oder das Ablaufdatum erreicht wurde
expireDate	dateTime	Datum (mit Uhrzeit), bis zu dem das WCM-Objekt gültig ist
keywords	String[]	Geordnete Menge von Schlüsselbegriffen, die diesem WCM-Objekt zugeordnet sind

Attribut	Datentyp	Beschreibung
`linkedFrom`	`ObjectId[]`	Array von OIDs der WCM-Objekte, die Referenzen auf das aktuelle WCM-Objekt enthalten. Falls keine Referenzen vorhanden sind, wird `null` zurückgeliefert.
`linksTo`	`Link[]`	Array der Referenzen, die sich im Inhalt des WCM-Objekts befinden. Falls keine Referenzen vorhanden sind, wird ein leeres Array zurückgeliefert.
`locale`	`Locale`	Für das WCM-Objekt verwendete Sprache (Locale)
`modifiedBy`	`User`	Kennung des Benutzers, der das WCM-Objekt zuletzt geändert hat
`modifiedDate`	`dateTime`	Datum (mit Uhrzeit), an dem das WCM-Objekt zuletzt geändert wurde
`objectCategory`	`String`	Name der Objektkategorie des WCM-Objekts oder `null`, wenn das WCM-Objekt keiner Objektkategorie zugeordnet ist
`objectId`	`ObjectId`	OID, die das WCM-Objekt innerhalb einer Website eindeutig identifiziert

Attribut	Datentyp	Beschreibung
`objectState`	`ObjectState`	Aktueller Bearbeitungsstatus des WCM-Objekts
`objectType`	`ObjectType`	Objekttyp des WCM-Objekts. Dieser kann im Rahmen der Regeln für kompatible Objekttypen geändert werden.
`pathname`	`String`	Pfadname der generierten Seite des WCM-Objekts. Dieser hängt vom Deploymentsystem ab. Wenn kein Deploymentsystem für diese Website existiert, steht hier `null`.
`pending ReleaseDate`	`dateTime`	Datum (mit Uhrzeit), an dem die Objektversion in der Produktionssicht veröffentlicht werden soll
`possible Actions`	`String[]`	Mögliche Staging-Aktionen, die mit dem aktuellen Status des WCM-Objekts und den Rechten des Benutzers durchgeführt werden können
`QAEMail Receivers`	`String[]`	E-Mail-Adressen, die zur Benachrichtigung verwendet werden sollen, wenn das WCM-Objekt zur Qualitätssicherung vorgelegt wurde

Attribut	Datentyp	Beschreibung
`releasedBy`	`User`	Kennung des Benutzers, der das WCM-Objekt zuletzt freigegeben hat
`releasedDate`	`dateTime`	Datum (mit Uhrzeit), an dem das WCM-Objekt zuletzt freigegeben wurde
`releaseEMail Receivers`	`String[]`	E-Mail-Adressen, die zur Benachrichtigung verwendet werden sollen, wenn das WCM-Objekt freigegeben wurde
`subtitle`	`String`	Überschrift des WCM-Objekts
`surrogateURL`	`String`	URL für die Surrogatseite des WCM-Objekts oder `null`, falls dieser Objekttyp keine Surrogatseite erfordert
`targetGroup`	`String`	Zielgruppe des WCM-Objekts
`template`	`ObjectId`	OID der Vorlage des WCM-Objekts oder `null`, falls dem Objekt keine Vorlage zugeordnet ist
`title`	`String`	Titel (Name) des WCM-Objekts

Attribut	Datentyp	Beschreibung
`topic`	`ObjectId`	OID des Themas, unterhalb dessen sich das WCM-Objekt befindet, oder `null`, falls das WCM-Objekt das Wurzelobjekt der Website ist
`URL`	`String`	URL für die generierte Seite des WCM-Objekts. Diese URL hängt vom Deploymentsystem ab, dem WCM WebServices zugeordnet wurde. Existiert kein Deployment-system, wird hier `null` geliefert.
`version`	`Version`	Aktuelle dreiteilige Versions-nummer des WCM-Objekts
`websiteName`	`String`	Name der Website, zu der das WCM-Objekt gehört

Datentyp: `ObjectType`

Allgemeine Informationen zu Objekttypen erhalten Sie im Abschnitt “Objekttyp” auf Seite 31.

Der Datentyp `ObjectType` repräsentiert den Typ eines WCM-Objekts. Objekttypen sind Website-spezifisch und nur innerhalb einer bestimmten Website gültig. Die Objekttypen einer Website können mit der Methode `getObjectTypes` ausgelesen werden (siehe “getObjectTypes” auf Seite 65). Mithilfe der Methode `change` kann der Objekttyp geändert werden (siehe “change” auf Seite 122). Eine Übersicht über kompatible Objekt-typen enthält das Attribut `compatibleObjectTypes` des Datentyps `ObjectData`.

Tabelle 14 – Die Komponenten des Datentyps `ObjectType`

Attribut	Datentyp	Beschreibung
`attributeSetName`	`String`	Name der Attributmenge, die diesem Objekttyp zugeordnet ist
`defaultSuffix`	`String`	Standard-Endung von Dateien, die Inhalte von WCM-Objekten dieses Typs enthalten. Diese Endung wird z. B. bei der Erzeugung neuer WCM-Objekte verwendet.
`description`	`String`	Lokalisierte Beschreibung des Objekttyps
`fileOnCreate Needed`	`boolean`	Dieser Wert ist genau dann wahr, wenn beim Anlegen eines WCM-Objekts dieses Typs eine Datei angegeben werden muss.
`frame`	`boolean`	Dieser Wert ist genau dann wahr, wenn der Objekttyp ein Frame ist.
`imageURL`	`String`	URL auf die Datei des Symbols, das den Objekttyp repräsentiert. Dabei ist die URL abhängig von dem Deploymentsystem, das WCM WebServices zugeordnet ist. Wenn kein Deploymentsystem für diese Website existiert, wird hier `null` geliefert.
`mimeType`	`String`	Zum Objekttyp gehöriger MIME-Typ
`name`	`String`	Name des Objekttyps
`template`	`boolean`	Dieser Wert ist genau dann wahr, wenn der Objekttyp eine Vorlage ist.
`topic`	`boolean`	Dieser Wert ist genau dann wahr, wenn der Objekttyp ein Thema ist.

Datentyp: `ObjectState`

Allgemeine Informationen zum Status von WCM-Objekten erhalten Sie im Abschnitt “Objektstatus” auf Seite 30.

Der Datentyp `ObjectState` repräsentiert den Status im Lebenszyklus eines WCM-Objekts.

Tabelle 15 – Die einzelnen Objektstatus

Status	Beschreibung
`checked_out`	Das WCM-Objekt wurde ausgeliehen.
`deleted`	Das WCM-Objekt wurde gelöscht.
`edited`	Das WCM-Objekt wurde geändert (Metadaten oder Inhalt).
`rejected`	Das WCM-Objekt wurde abgelehnt.
`released`	Das WCM-Objekt wurde freigegeben.
`release_at`	Das WCM-Objekt wurde freigegeben, wird jedoch erst später veröffentlicht.
`submitted`	Das WCM-Objekt wurde der Qualitätssicherung vorgelegt.

Die folgende Tabelle zeigt die Status und die möglichen Übergänge zwischen den Status. Der Übergang zu einem anderen Status erfolgt im Rahmen von Staging-Aktionen. Wenn ein WCM-Objekt neu angelegt wird, erhält es den Status “edited”. Informationen zu den einzelnen Staging-Methoden erhalten Sie im Abschnitt 5.3 “Staging-Methoden” ab Seite 100.

Tabelle 16 – Die möglichen Übergänge zwischen den Objektstatus

Status vor der Aktion	Status nach der Aktion	Aktion
edited	edited	change
	checked_out	checkOut
	deleted	delete
	submitted	submit
checked_out	edited	checkIn
	Zurück zum Status, den das Objekt vorher hatte	undoCheckOut
deleted	Das Objekt wird endgültig aus dem WCM-System entfernt.	destroy
	rejected	reject
submitted	rejected	reject
	released	release
	pending_release	release
pending_release	released	Automatisch, wenn das Veröffentlichungsdatum erreicht ist
	checked_out	checkOut
	edited	change
released	checked_out	checkOut
	edited	change

Tabelle 17 – Die Komponenten des Datentyps `ObjectState`

Attribut	Datentyp	Beschreibung
`description`	`String`	Lokalisierte Beschreibung des Objektstatus
`imageURL`	`String`	URL auf die Datei des Symbols, das den Objektstatus repräsentiert. Dabei ist die URL abhängig vom Deploymentsystem, das WCM WebServices zugeordnet ist. Wenn kein Deploymentsystem für diese Website existiert, wird hier `null` geliefert.
`name`	`String`	Name des Status

Datentyp: `Permission`

Allgemeine Informationen zu Zugriffsrechten erhalten Sie im Abschnitt “Zugriffsrechte für WCM-Objekte” auf Seite 35.

Der Datentyp `Permission` repräsentiert ein Zugriffsrecht, das für WCM-Objekte gewährt oder verweigert werden kann. Jedes WCM-Objekt hat eine Zugriffssteuerungsliste (ACL), die eine Menge von Principals zusammen mit ihren gewährten und verweigerten Zugriffsrechten enthält. Das WCM-Objekt erbt die ACL des übergeordneten Objekts (rekursiv), wenn es keine eigene ACL hat. Weitere Informationen zu ACLs erhalten Sie im Abschnitt “Datentyp: `Acl`” auf Seite 90.

Tabelle 18 – Die möglichen Zugriffsrechte für WCM-Objekte

Name	Abkürzung	Beschreibung
CHANGE_RIGHTS	cr	Recht zur Änderung der Zugriffsrechte eines WCM-Objekts
CREATE	c	Recht zum Anlegen eines neuen WCM-Objekts
DELETE	d	Recht zum Löschen eines WCM-Objekts
READ	r	Recht zum Lesen eines WCM-Objekts
READ_PRODUCTION	rp	Recht zum Lesen eines WCM-Objekts in der Produktionssicht
RELEASE	rl	Recht zum Freigeben eines WCM-Objekts
TREE_OPERATIONS	t	Recht zum Kopieren und Verschieben eines WCM-Objekts
WRITE	w	Recht zum Ändern eines WCM-Objekts
WRITE_META	wm	Recht zum Ändern von Metadaten

Tabelle 19 – Die Komponenten des Datentyps `Permission`

Attribut	Datentyp	Beschreibung
name	String	Name (als Abkürzung) des Zugriffsrechts

Datentypen für Zugriffssteuerungslisten

Für das Bearbeiten von Zugriffssteuerungslisten stehen zwei Datentypen zur Verfügung:

- `Acl`
- `AclEntry`

Datentyp: `Acl`

Eine Zugriffssteuerungsliste (Access Control List = ACL) kombiniert eine Menge von Zugriffsrechten. In einem WCM-System enthält eine ACL typischerweise alle Zugriffsrechte, die für ein WCM-Objekt gesetzt wurden.

In einer ACL sind die Zugriffsrechte für Principals (Benutzer, Gruppen, Rollen, Gruppenrollen oder “Jeder”) gesetzt. Ein Zugriffsrecht kann gewährt oder explizit verweigert sein.

Tabelle 20 – Die Komponenten des Datentyps `Acl`

Attribut	Datentyp	Beschreibung
`entries`	`AclEntry[]`	Array von Zugriffsrechten, die dem gegebenen Principal gewährt oder verweigert werden
`inherited`	`boolean`	Dieser Wert ist genau dann wahr, wenn dies ein geerbtes ACL-Objekt ist, d. h. wenn die Zugriffsrechte vom übergeordneten Objekt übernommen wurden. Das Objekt, zu dem die geerbte ACL gehört, ist über das Attribut `owner` verfügbar.
`owner`	`ObjectId`	OID des WCM-Objekts, zu dem diese ACL gehört

Datentyp: AclEntry

Dieser Datentyp repräsentiert die Zuordnung eines Principals zu einem gewährten oder verweigerten Zugriffsrecht.

Tabelle 21 – Die Komponenten des Datentyps AclEntry

Attribut	Datentyp	Beschreibung
permission	Permission	Zugriffsrecht, das dem Principal von diesem AclEntry zugewiesen wird
policy	boolean	Definiert, ob dem Principal das Zugriffsrecht gewährt (wahr) oder verweigert (falsch) werden soll
principal	Principal	Zugehöriger Principal

ACL eines WCM-Objekts ändern

So ändern Sie die ACL eines WCM-Objekts:

1. Erzeugen Sie für jeden Principal, der in der neuen ACL vorkommen soll, ein entsprechendes User-, Group-, Role- bzw. GroupRole-Objekt.
2. Erzeugen Sie für jede Rechteart, die in der ACL definiert werden soll, ein entsprechendes Permission-Objekt.
3. Erzeugen Sie mit den angelegten Principal- und Permission-Objekten für jedes Recht, das gewährt bzw. verweigert werden soll, ein AclEntry-Objekt. In diesem AclEntry-Objekt wird über das Attribut policy gesteuert, ob das Recht gewährt (true) oder verweigert (false) wird.
4. Legen Sie ein Array von AclEntries an, in das die gerade erzeugten AclEntries eingetragen werden.

5. Mit diesem Feld wird das `entries`-Attribut eines `Acl`-Objekts gesetzt, das `owner`- und `inherited`-Attribut spielen dabei keine Rolle.
6. Belegen Sie folgende Attribute eines `ObjectData`-Objekts mit folgenden Werten:
   - das `ACL`-Attribut mit dem erzeugten `Acl`-Objekt
   - die `ObjectId` mit der OID des zu ändernden WCM-Objekts
   - das Array `attributeKeys` mit dem Attributnamen `ACL`
7. Rufen Sie mit diesem `ObjectData`-Objekt die Methode `change` auf, um die neuen Rechte für das WCM-Objekt zu setzen, siehe “change” auf Seite 122.

Datentyp: `MultiImportPart`

Dieser Datentyp dient dazu, eine baumartige Struktur von WCM-Objekten zu repräsentieren, die von Livelink WCM Server mit der Methode `multiImport` angelegt werden können (siehe “multiImport” auf Seite 142).

Tabelle 22 – Die Komponenten des Datentyps `MultiImportPart`

Attribut	Datentyp	Beschreibung
`children`	`MultiImportPart[]`	Hier werden die unterhalb des aktuellen WCM-Objekts anzulegenden WCM-Objekte definiert. Diese Liste wird nur ausgewertet, wenn das anzulegende Objekt den Typ “Thema” hat.

Attribut	Datentyp	Beschreibung
`content`	`base64Binary`	Inhalt des anzulegenden WCM-Objekts. Hier darf `null` stehen, wenn das Objekt ohne Inhalt angelegt werden soll. Hinsichtlich des Inhalts gelten die Regeln für den jeweiligen Objekttyp: So muss ein Objekt vom Typ “Frame” zum Beispiel einen `frameset`-Tag enthalten.
`objectData`	`ObjectData`	Hier werden die Metadaten des anzulegenden Objekts definiert. Folgende Felder müssen dabei gesetzt sein: `deploymentHint`: Der Vorschlag für den Dateinamen der generierten Seite `objectType`: Der Objekttyp des anzulegenden WCM-Objekts `title`: Der Titel des WCM-Objekts `attributeKeys`: Die Liste der Namen der gesetzten Attribute (mindestens `deploymentHint`, `objectType`, `title`)

Datentypen für Workflow-Aktionen

In Livelink WCM Server können die vordefinierten Staging-Stufen (Bearbeitung, Qualitätssicherung, Veröffentlichung im Produktionsbetrieb) um eigene Workflow-Schritte erweitert werden, um z. B. eine Bearbeitung durch mehrere Redakteure oder eine mehrstufige Qualitätssicherung durchzuführen.

Jede Workflow-Definition besteht aus beliebig vielen Aktivitäten und Übergängen. Einer Aktivität ist jeweils ein Principal (Benutzer, Gruppe oder Rolle) zugeordnet, der das Objekt bearbeiten und im Workflow weiterleiten darf. Die drei Aktivitätstypen "Bearbeiten", "QS" und "Löschen" repräsentieren die Aufgabe des zugeordneten Principals. Mithilfe der Übergänge wird das WCM-Objekt von einer Aktivität zur nächsten weitergeleitet. Die notwendigen Staging-Übergänge werden dabei automatisch im Hintergrund ausgeführt.

Für das Arbeiten mit Content Workflow stehen vier Datentypen zur Verfügung:

- `Workflow`
- `Activity`
- `Transition`
- `WorkflowNavigationInfo`

Datentyp: `Workflow`

Ein Objekt vom Typ "Workflow" repräsentiert eine so genannte Workflow-Definition, in der die einzelnen Workflow-Schritte festgelegt sind. Workflow-Definitionen werden über den Content Workflow Modeler angelegt und bearbeitet. Zum Anlegen von Objekten vom Typ "Workflow" benötigt ein Benutzer den Funktionsbereich "Workflow".

Tabelle 23 – Die Komponenten des Datentyps `Workflow`

Attribut	Datentyp	Beschreibung
`activities`	`Activity[]`	Einzelne Aktivitäten eines Workflows
`assignedObject`	`ObjectId`	Dem Workflow zugeordnetes WCM-Objekt
`currentActivities`	`Activity[]`	Alle Aktivitäten eines Workflows
`currentActivity`	`Activity`	Aktuelle Aktivität
`description`	`String`	Beschreibung des Workflow-Objekts
`hasBeenForwarded`	`Boolean`	Ermittelt, ob das zugeordnete Workflow-Objekt bereits weitergeschaltet wurde
`hasContent`	`Boolean`	Ermittelt, ob die XPDL-Datei, auf der die Workflow-Definition basiert, über Inhalt verfügt
`name`	`String`	Name des Workflow-Objekts
`processId`	`String`	Prozess-ID
`startActivities`	`Activity[]`	Alle Start-Aktivitäten des aktuellen Workflow-Objekts
`workflowOid`	`ObjectId`	OID des Workflow-Objekts
`workflowTitle`	`String`	Titel des Workflow-Objekts
`workflowVersion`	`Version`	Version des Workflow-Objekts

Datentyp: `Activity`

Die einzelnen Aktivitäten eines Workflows werden durch den Datentyp `Activity` repräsentiert. Die Aktivitäten haben einen Typ, der das Staging in Livelink WCM Server widerspiegelt. Weiterführende Informationen zur Verwendung und Nutzung von Aktivitäten finden Sie im WCM Java API-Programmierhandbuch.

Tabelle 24 – Die Komponenten des Datentyps `Activity`

Attribut	Datentyp	Beschreibung
`activityType`	`String`	Staging-Typ der Aktivität
`description`	`String`	Beschreibung der Aktivität
`id`	`String`	Aktivitäts-ID
`name`	`String`	Name der Aktivität
`principalName`	`String`	Name des Principals, der der Aktivität zugeordnet ist
`principalType`	`int`	Typ des Principals, der der Aktivität zugeordnet ist
`transitions`	`Transition[]`	Alle von der Aktivität wegführenden Übergänge

Datentyp: `Transition`

Ein gerichteter Übergang zwischen zwei Aktivitäten eines Workflows wird durch das Interface `Transition` repräsentiert. Neben der Bezeichnung und der Beschreibung können zu einem Übergang die zugeordneten Aktivitäten abgefragt werden.

Tabelle 25 – Die Komponenten des Datentyps `Transition`

Attribut	Datentyp	Beschreibung
`description`	`String`	Beschreibung des Übergangs
`fromActivity`	`Activity`	Aktivität, die Ausgangspunkt des Übergangs ist
`id`	`String`	Aktivitäts-ID
`toActivity`	`Activity`	Aktivität, die Zielpunkt des Übergangs ist
`name`	`String`	Name des Übergangs

Datentyp: `WorkflowNavigationInfo`

Das `WorkflowNavigationInfo`-Objekt übernimmt die Funktion eines Containers, der Informationen über Workflows zusammenfasst. Derartige Informationen können für die Entwicklung von Benutzeroberflächen für verwendete Workflows eingesetzt werden.

Tabelle 26 – Die Komponenten des Datentyps `WorkflowNavigationInfo`

Attribut	Datentyp	Beschreibung
`activity`	`Activity`	Mögliche Aktivitäten eines Principals
`workflow`	`Workflow`	Workflow-Definition

5.2 Allgemeine Parameter

Einige der im Folgenden beschriebenen Methoden verwenden die Parameter `DeploymentWaitInfo` und `EmailWaitInfo`. Diese Parameter haben immer die gleiche Funktion und werden daher nur einmal an dieser Stelle beschrieben.

Parameter `DeploymentWaitInfo`

Der Datentyp `DeploymentWaitInfo` repräsentiert die Informationen, die verwendet werden, um auf alle Jobs der Deploymentsysteme zu warten, die zu einer Aktion gehören. Dieser Datentyp wird in Prozeduraufrufen verwendet, die Staging-Aktionen anstoßen.

Ist der Parameter `DeploymentWaitInfo` nicht `null`, so wartet die aufrufende Methode darauf, dass die in diesem Parameter spezifizierten Deploymentsysteme alle zugehörigen Deploymentaufgaben abschließen. Die maximale Wartezeit wird durch `DeploymentWaitInfo.timeout` definiert. Die Methode kehrt spätestens zurück, wenn die Wartezeit für diese Aktionen den angegebenen Timeout-Wert überschreitet.

Wenn im Aufruf einer Staging-Methode der Parameter für `DeploymentWaitInfo` gleich `null` ist, wird auf kein Deploymentsystem gewartet. Die Abarbeitung der Deploymentaufgaben geschieht dann asynchron.

Tabelle 27 – Die Komponenten des Datentyps `DeploymentWaitInfo`

Attribut	Datentyp	Beschreibung
`deploymentSystems`	`String[]`	Liste von Deploymentsystemen, auf die gewartet werden soll
`isTimeout`	`boolean`	In WCM WebServices ist dies ein Feld, das für zukünftige Funktionen reserviert ist. Zum jetzigen Zeitpunkt hat es keine Bedeutung.
`timeout`	`long`	Maximale Zeit (in Millisekunden), die auf die Beendigung der Deploymentaufgaben gewartet wird

Parameter `EMailInfo`

Der Datentyp `EMailInfo` repräsentiert die Informationen, die zum automatischen Erstellen von E-Mails verwendet werden. Diese E-Mails werden beim Vorlegen, Freigeben, Löschen und Ablehnen von WCM-Objekten versendet. Das Senden der E-Mails kann auf dem WCM-Server erfolgen, wenn die entsprechenden Methoden verwendet werden.

Der im WCM-Server implementierte Algorithmus sammelt alle E-Mail-Adressen aus den Daten des WCM-Objekts. Die Aktion definiert, welche Attribute zur Erzeugung der E-Mail-Adressen verwendet werden. Die relevanten Attribute sind `editEMailReceivers`, `QAEMailReceivers` und `releaseEMailReceivers`.

Wenn im Aufruf einer Staging-Methode der Parameter für die `EMailInfo` gleich null ist, erzeugt das WCM-System keine E-Mails.

Tabelle 28 – Die Komponenten des Datentyps `EMailInfo`

Attribut	Datentyp	Beschreibung
`cc`	`String`	“CC”-Feld der E-Mail. Hier können zusätzliche Empfänger eingetragen werden.
`excludeOIDList`	`boolean`	Dieser Wert ist genau dann wahr, wenn die Liste der OIDs in der erzeugten E-Mail weggelassen werden soll.
`subject`	`String`	Betreff der E-Mail
`text`	`String`	Inhalt der E-Mail

5.3 Staging-Methoden

In diesem Abschnitt werden die Methoden von WCM WebServices vorgestellt, mit denen Staging-Aktionen realisiert werden können.

- Ausleihen von Objekten zur Bearbeitung, siehe “checkOut” auf Seite 102
- Ausleihen von Objekten rückgängig machen, siehe “undoCheckOut” auf Seite 108
- Zurückgeben von Objekten nach der Bearbeitung, siehe “checkIn” auf Seite 101
- Objekte ohne Vorlegen direkt freigeben, siehe “directRelease” auf Seite 103
- Objekte zur Qualitätssicherung vorlegen, siehe “submit” auf Seite 106 und “submitImmediately” auf Seite 107

- Objekte in der Qualitätssicherung ablehnen, siehe “reject” auf Seite 104
- Objekte in der Qualitätssicherung freigeben, siehe “release” auf Seite 105

In der folgenden Übersicht sind die Methoden alphabetisch sortiert.

checkIn

Gibt den (modifizierten) Inhalt des angegebenen WCM-Objekts an das WCM-System zurück

Parameter

Tabelle 29 – Parameter der Methode `checkIn`

Parameter	Datentyp	Beschreibung
`oid`	`ObjectId`	OID des betreffenden WCM-Objekts
`content`	`byte[]`	Inhalt des WCM-Objekts, als Datei spezifiziert
`keepCheckOut`	`boolean`	Flagge, ob das Objekt nach dem Aufruf weiterhin ausgeliehen bleiben soll
`remark`	`String`	Zeichenkette, die zum Protokoll des WCM-Objekts hinzugefügt wird, oder `null`
`dplWaitInfo`	`DeploymentWaitInfo`	Liste der Deploymentsysteme oder `null`

Rückgabe

keine

Voraussetzungen

- Erforderliche Zugriffsrechte: `READ` und `WRITE`
- Objektstatus: muss `CHECKED_OUT` sein

Folge

Nach dem Zurückgeben hat das WCM-Objekt den Status

- `EDITED`, falls `keepCheckOut` den Wert `false` hatte
- `CHECKED_OUT`, falls `keepCheckOut` den Wert `true` hatte

checkOut

Leiht das angegebene WCM-Objekt aus

Parameter

Tabelle 30 – Parameter der Methode `checkOut`

Parameter	Datentyp	Beschreibung
`oid`	`ObjectId`	OID des betreffenden WCM-Objekts
`remark`	`String`	Zeichenkette, die zum Protokoll des WCM-Objekts hinzugefügt wird, oder `null`
`dplWaitInfo`	`DeploymentWaitInfo`	Liste der Deploymentsysteme oder `null`

Rückgabe

keine

Voraussetzungen

- Erforderliche Zugriffsrechte: `READ` und `WRITE`
- Objektstatus: muss `EDITED`, `REJECTED`, `RELEASED` oder `PENDING_RELEASE` sein

Folge

Nach dem Ausleihen hat das WCM-Objekt den Status `CHECKED_OUT`.

> **Hinweis:** Um den Objektinhalt zu bearbeiten, muss anschließend `getCheckOutContent` aufgerufen werden, siehe “getCheckOutContent” auf Seite 134.

`directRelease`

Gibt eine Liste von WCM-Objekten ohne eine Vorlage zur Qualitätssicherung frei

Parameter

Tabelle 31 – Parameter der Methode `directRelease`

Parameter	Datentyp	Beschreibung
`oids`	`ObjectId[]`	OIDs der WCM-Objekte, die direkt freigegeben werden
`pendingReleaseDate`	`Date`	Datum und Uhrzeit, wann die angegebenen WCM-Objekte freigegeben werden sollen (verzögerte Freigabe)
`remark`	`String`	Zeichenkette, die zum Protokoll der WCM-Objekte hinzugefügt wird, oder `null`
`emailInfo`	`EMailInfo`	E-Mail-Information, die an die in den Objektdaten definierten Empfänger gesendet wird, oder `null`, falls keine E-Mail gesendet werden soll
`dplWaitInfo`	`DeploymentWaitInfo`	Liste der Deploymentsysteme oder `null`

Rückgabe

keine

Voraussetzungen

- Erforderliche Zugriffsrechte: `READ` und `WRITE` oder `WRITE_META`
- Objektstatus: muss `EDITED` sein

Folge

Nach der direkten Freigabe hat WCM-Objekt den Status `RELEASED` bzw. `PENDING_RELEASE` (falls ein Freigabetermin in der Zukunft angegeben wurde).

`reject`

Lehnt eine Liste von WCM-Objekten ab

Parameter

Tabelle 32 – Parameter der Methode `reject`

Parameter	Datentyp	Beschreibung
`oids`	`ObjectId[]`	OIDs der WCM-Objekte, die abgelehnt werden
`remark`	`String`	Zeichenkette, die zum Protokoll jedes WCM-Objekts hinzugefügt wird, oder `null`
`emailInfo`	`EMailInfo`	E-Mail-Information, die an die in den Objektdaten definierten Empfänger gesendet wird, oder `null`, falls keine E-Mail gesendet werden soll
`dplWaitInfo`	`DeploymentWaitInfo`	Liste der Deploymentsysteme oder `null`

Rückgabe

keine

Voraussetzungen

- Erforderliche Zugriffsrechte: `READ` und `RELEASE`
- Objektstatus: muss `SUBMITTED` oder `DELETED` sein

Folge

Nach dem Ablehnen haben die WCM-Objekte den Status `REJECTED`.

`release`

Gibt eine Liste von WCM-Objekten frei

Parameter

Tabelle 33 – Parameter der Methode `release`

Parameter	Datentyp	Beschreibung
`oids`	`ObjectId[]`	OIDs der WCM-Objekte, die freigegeben werden
`remark`	`String`	Zeichenkette, die zum Protokoll jedes WCM-Objekts hinzugefügt wird, oder `null`
`emailInfo`	`EMailInfo`	E-Mail-Information, die an die in den Objektdaten definierten Empfänger gesendet wird, oder `null`, falls keine E-Mail gesendet werden soll
`dplWaitInfo`	`DeploymentWaitInfo`	Liste der Deploymentsysteme oder `null`

Rückgabe

keine

Voraussetzungen

- Erforderliche Zugriffsrechte: `READ` und `RELEASE`
- Objektstatus: muss `SUBMITTED` sein

Folge

Nach der Freigabe haben die WCM-Objekte den Status `RELEASED` oder `PENDING_RELEASE`. Der Statuswechsel bei der Freigabe hängt davon ab, ob das Metadatum “Verzögerte Freigabe” gesetzt ist.

`submit`

Legt eine Liste von WCM-Objekten zur Qualitätssicherung vor

Parameter

Tabelle 34 – Parameter der Methode `submit`

Parameter	Datentyp	Beschreibung
`oids`	`ObjectId[]`	OIDs der WCM-Objekte, die vorgelegt werden
`pendingReleaseDate`	`Date`	Datum und Uhrzeit bei verzögerter Freigabe oder `null`, falls eine sofortige Freigabe ermöglicht werden soll
`remark`	`String`	Zeichenkette, die zum Protokoll der WCM-Objekte hinzugefügt wird, oder `null`
`emailInfo`	`EMailInfo`	E-Mail-Information, die an die in den Objektdaten definierten Empfänger gesendet wird, oder `null`, falls keine E-Mail gesendet werden soll
`dplWaitInfo`	`DeploymentWaitInfo`	Liste der Deploymentsysteme oder `null`

Rückgabe

keine

Voraussetzungen

- Erforderliche Zugriffsrechte: `READ` und `WRITE` oder `WRITE_META`
- Objektstatus: muss `EDITED` oder `REJECTED` sein

Folge

Nach dem Vorlegen haben die WCM-Objekte den Status `SUBMITTED`.

`submitImmediately`

Legt eine Liste von WCM-Objekten zur Qualitätssicherung vor und ermöglicht eine sofortige Freigabe

Parameter

Tabelle 35 – Parameter der Methode `submitImmediately`

Parameter	Datentyp	Beschreibung
`oids`	`ObjectId[]`	OIDs der WCM-Objekte, die vorgelegt werden
`remark`	`String`	Zeichenkette, die zum Protokoll der WCM-Objekte hinzugefügt wird, oder `null`
`emailInfo`	`EMailInfo`	E-Mail-Information, die an die in den Objektdaten definierten Empfänger gesendet wird, oder `null`, falls keine E-Mail gesendet werden soll
`dplWaitInfo`	`DeploymentWaitInfo`	Liste der Deploymentsysteme oder `null`

Rückgabe

keine

Voraussetzungen

- Erforderliche Zugriffsrechte: `READ` und `WRITE` oder `WRITE_META`
- Objektstatus: muss `EDITED` oder `REJECTED` sein

Folge

Nach dem Vorlegen haben die WCM-Objekte den Status `SUBMITTED`.

undoCheckOut

Das Ausleihen der angegebenen WCM-Objekte wird rückgängig gemacht. Dadurch sind die Objekte zur Bearbeitung durch andere verfügbar.

Parameter

Tabelle 36 – Parameter der Methode `undoCheckOut`

Parameter	Datentyp	Beschreibung
`oids`	`ObjectId[]`	OIDs der betreffenden WCM-Objekte
`remark`	`String`	Zeichenkette, die zum Protokoll der WCM-Objekte hinzugefügt wird, oder `null`
`dplWaitInfo`	`DeploymentWaitInfo`	Liste der Deploymentsysteme oder `null`

Rückgabe

keine

Voraussetzungen

- Erforderliche Zugriffsrechte: `READ` und `WRITE`
- Objektstatus: muss `CHECKED_OUT` sein

Folge

Nachdem die Änderungen an den ausgeliehenden WCM-Objekten verworfen wurden, haben die WCM-Objekte den Status `EDITED`.

5.4 Workflow-Methoden

Die Schnittstelle für die Verwaltung von Workflow-Objekten erlaubt das Zuordnen eines Workflows zu einem WCM-Objekt, das Weiterleiten von WCM-Objekten im Workflow und das Aufheben der Zuordnung zwischen Workflow- und WCM-Objekt.

In diesem Abschnitt werden die Methoden von WCM WebServices vorgestellt, mit denen die Workflow-Aktionen realisiert werden können.

`assignWorkflow`

Ordnet die aktuell freigegebene Version eines Workflow-Objekts dem angegebenen WCM-Objekt zu. Ein initialer Übergang muss ausgewählt werden, der nach der Zuordnung automatisch ausgeführt wird.

Parameter

Tabelle 37 – Parameter der Methode `assignWorkflow`

Parameter	Datentyp	Beschreibung
`oid`	`ObjectId`	OID des WCM-Objekts, dem ein Workflow zugeordnet werden soll
`workflowOid`	`ObjectId`	OID des zugeordneten Workflows

Parameter	Datentyp	Beschreibung
`startTransition`	`String`	Name des initialen Übergangs
`remark`	`String`	Zeichenkette, die zum Protokoll des WCM-Objekts hinzugefügt wird, oder `null`
`emailInfo`	`EMailInfo`	E-Mail-Information, die an den Principal gesendet wird, der der nächsten Aktivität zugeordnet ist, oder `null`, falls keine E-Mail gesendet werden soll

Rückgabe

keine

Voraussetzungen

- Erforderliche Zugriffsrechte: `READ`
- Objektstatus: wird durch die initiale Workflow-Aktivität bestimmt
- Funktionsbereich: `WORKFLOW`

Folge

Der Objektstatus ändert sich nicht.

`assignWorkflowAsync`

Ordnet die aktuell freigegebene Version eines Workflow-Objekts **mehreren** WCM-Objekten zu. Ein initialer Übergang muss ausgewählt werden, der nach der Zuordnung automatisch ausgeführt wird. Bei dieser Methode handelt es sich um eine asynchrone Methode, d.h., die Methode kehrt praktisch sofort nach dem Aufruf zurück und wartet nicht, bis die damit verbundenen Aufgaben abgeschlossen sind.

Parameter

Tabelle 38 – Parameter der Methode `assignWorkflowAsync`

Parameter	Datentyp	Beschreibung
`oids`	`ObjectId[]`	OIDs der WCM-Objekte, denen ein Workflow zugeordnet werden soll
`workflowOid`	`ObjectId`	OID des zugeordneten Workflows
`startTransition`	`String`	Name des initialen Übergangs
`remark`	`String`	Zeichenkette, die zum Protokoll der WCM-Objekte hinzugefügt wird, oder `null`
`emailInfo`	`EMailInfo`	E-Mail-Information, die an den Principal gesendet wird, der der nächsten Aktivität zugeordnet ist, oder `null`, falls keine E-Mail gesendet werden soll

Rückgabe

keine

Voraussetzungen

- Erforderliche Zugriffsrechte: `READ`
- Objektstatus: wird durch die initiale Workflow-Aktivität bestimmt
- Funktionsbereich: `WORKFLOW`

Folge

Der Objektstatus ändert sich nicht.

forward

Leitet ein Objekt im Workflow über den angegebenen Übergang von der aktuellen zur nächsten Aktivität weiter

Parameter

Tabelle 39 – Parameter der Methode `forward`

Parameter	Datentyp	Beschreibung
`workflow`	`Workflow`	Workflow-Definition
`transition`	`Transition`	Übergang, über den auf die nächste Aktivität verwiesen wird
`remark`	`String`	Zeichenkette, die zum Protokoll des WCM-Objekts hinzugefügt wird, oder `null`
`emailInfo`	`EMailInfo`	E-Mail-Information, die an den Principal gesendet wird, der der nächsten Aktivität zugeordnet ist, oder `null`, falls keine E-Mail gesendet werden soll

Rückgabe

keine

Voraussetzungen

Erforderliche Zugriffsrechte und Funktionsbereiche werden durch die Workflow-Aktivität bestimmt.

Folge

Der Objektstatus wird durch die letzte Workflow-Aktivität bestimmt.

getAssignedJobs

Liefert die IDs aller WCM-Objekte zurück, denen ein bestimmer Workflow zugeordnet ist und die sich in einer Aktivität befinden, der der angemeldete Principal zugeordnet ist

Parameter

Tabelle 40 – Parameter der Methode `getAssignedJobs`

Parameter	Datentyp	Beschreibung
`workflow`	`Workflow`	Workflow-Definition oder `null`
`activityId`	`String`	Aktivitäts-ID oder `null`

Rückgabe

`ObjectId[] objectIds`: Liste von OIDs

Voraussetzungen

keine

Folge

Der Objektstatus ändert sich nicht.

getAssignedWorkflow

Liefert für das angegebene WCM-Objekt die zugeordnete Workflow-Instanz

Tabelle 41 – Parameter der Methode `getAssignedWorkflow`

Parameter	Datentyp	Beschreibung
`oid`	`ObjectId`	OID des WCM-Objekts, dessen zugeordneter Workflow ermittelt werden soll

Rückgabe

`Workflow workflow`: das aktuell zugeordnete Workflow-Objekt

Voraussetzungen

Keine

Folge

Der Objektstatus ändert sich nicht.

`getAssignedWorkflowActivities`

Liefert eine Liste von `WorkflowNavigationInfo`-Objekten, deren Aktivitäten vom angemeldeten Principal ausgeführt werden können

Rückgabe

`WorkflowNavigationInfo[] workflowNavigationInfos`: eine Menge von `WorkflowNavigationInfo`-Objekten

Voraussetzungen

Keine

Folge

Der Objektstatus ändert sich nicht.

`getAssignedWorkflows`

Liefert alle Workflow-Objekte, deren aktueller Aktivität der angemeldete Principal zugeordnet ist

Rückgabe

`Workflow[] workflows`: eine Menge von Workflow-Objekten

Voraussetzungen

Keine

Folge

Der Objektstatus ändert sich nicht.

getAvailableWorkflows

Liefert alle Workflow-Objekte, für die der Principal über das Zugriffsrecht READ verfügt

Rückgabe

`Workflow[] workflows`: eine Menge von Workflow-Objekten

Voraussetzungen

- Erforderliches Zugriffsrecht: READ
- Objektstatus: keine Einschränkungen

Folge

Der Objektstatus ändert sich nicht.

getWorkflow

Liefert die Workflow-Definition in der angegebenen Version

Tabelle 42 – Parameter der Methode getWorkflow

Parameter	Datentyp	Beschreibung
workflowOid	ObjectId	OID des Workflow-Objekts
version	Version	Version des Workflow-Objekts

Rückgabe

`Workflow workflow`: ein Workflow-Objekt

Voraussetzungen

- Erforderliches Zugriffsrecht: READ
- Objektstatus: keine Einschränkungen

Folge

Der Objektstatus ändert sich nicht.

isForwardable

Überprüft, ob der angemeldete Principal der Aktivität, in der sich das WCM-Objekt aktuell befindet, zugeordnet ist

Tabelle 43 – Parameter der Methode `isForwardable`

Parameter	Datentyp	Beschreibung
`oid`	`ObjectId`	OID des WCM-Objekts mit zugeordnetem Workflow

Rückgabe

`true`, wenn der Principal das Objekt im Workflow weiterleiten darf, sonst `false`

Voraussetzungen

Keine

Folge

Der Objektstatus ändert sich nicht.

isRemovable

Überprüft, ob der angemeldete Principal die Zuordnung zwischen Workflow- und WCM-Objekt aufheben darf

Tabelle 44 – Parameter der Methode `isRemovable`

Parameter	Datentyp	Beschreibung
`oid`	`ObjectId`	OID des Workflow-Objekts

Rückgabe

`true`, wenn der Principal die Zuordnung zwischen Workflow- und WCM-Objekt aufheben darf, sonst `false`

Voraussetzungen

Keine

Folge

Der Objektstatus ändert sich nicht.

`removeWorkflow`

Hebt die Zuordnung zwischen Workflow- und WCM-Objekt auf

Tabelle 45 – Parameter der Methode `removeWorkflow`

Parameter	Datentyp	Beschreibung
`oid`	`ObjectId`	OID des WCM-Objekts, dessen Zuordnung zu einem Workflow-Objekt aufgehoben werden soll
`remark`	`String`	Zeichenkette, die zum Protokoll des WCM-Objekts hinzugefügt wird, oder `null`

Rückgabe

Keine

Voraussetzungen

- Erforderliches Zugriffsrecht: `READ` und `CHANGE_RIGHTS`
- Objektstatus: keine Einschränkungen

Folge

Der Objektstatus ändert sich nicht.

removeWorkflowAsync

Hebt die Zuordnung zwischen dem Workflow-Objekt und **mehreren** WCM-Objekten auf. Hierbei handelt es sich um eine asynchrone Methode, d.h., die Methode kehrt praktisch sofort nach dem Aufruf zurück und wartet nicht, bis die damit verbundenen Aufgaben abgeschlossen sind.

Tabelle 46 – Parameter der Methode `removeWorkflowAsync`

Parameter	Datentyp	Beschreibung
`oids`	`ObjectId[]`	OIDs der WCM-Objekte, deren Zuordnung zu einem Workflow-Objekt aufgehoben werden soll
`remark`	`String`	Zeichenkette, die zum Protokoll der WCM-Objekte hinzugefügt wird, oder `null`

Rückgabe

Keine

Voraussetzungen

- Erforderliches Zugriffsrecht: `READ` und `CHANGE_RIGHTS`
- Objektstatus: keine Einschränkungen

Folge

Der Objektstatus ändert sich nicht.

5.5 Methoden der Objektverwaltung

In diesem Abschnitt werden die Methoden von WCM WebServices vorgestellt, mit denen WCM-Objekte bearbeitet werden können.

Objekte anlegen

- neue Objekte anlegen, siehe “create” auf Seite 128
- verknüpfte Objekte durch einen Import anlegen, siehe “multiImport” auf Seite 142

Objekte erhalten und bearbeiten

- Objekt mit der angegebenen OID in der definierten Version lesen, siehe “get” auf Seite 133
- Inhalt des WCM-Objekts mit der angegebenen OID und der definierten Version erhalten. Der Inhalt wird vom zugeordneten Deploymentsystem generiert. Dabei wird der Inhalt mit den verwendeten Vorlagen zusammengeführt. Siehe “getCheckOutContent” auf Seite 134.
- Inhalt des WCM-Objekts mit der angegebenen OID und der definierten Version erhalten, ohne weitere Manipulationen am Inhalt in Bezug auf die verwendete Vorlage. Siehe “getContent” auf Seite 136.
- Metadaten ändern, siehe “change” auf Seite 122
- Verbundobjekte konvertieren, siehe “convertContent” auf Seite 126

Position in der Hierarchie ermitteln/sortieren

- Ermitteln der untergeordneten Objekte eines Themas, siehe “getChildren” auf Seite 135
- Ermitteln des übergeordneten Objekts, siehe “getParent” auf Seite 139

- Sortierung von Elementen, sodass jedes Kind-Objekt hinter seinem Eltern-Objekt platziert ist, siehe “sortParentsFirst” auf Seite 146

Objekte kopieren und verschieben

- Objekte kopieren, siehe “copy” auf Seite 127
- Objekt verschieben, siehe “move” auf Seite 141

Objekte löschen

- Objekte löschen (muss von der Qualitätssicherung bestätigt werden), siehe “delete” auf Seite 129
- Objekte endgültig löschen, siehe “destroy” auf Seite 131

Protokoll und Versionen

- Abfragen von Protokolleinträgen, siehe “getLastLogEntries” auf Seite 138
- Bemerkung zum Protokoll hinzufügen, siehe “addRemark” auf Seite 121
- Versionen eines Objekts auflisten, siehe “getVersionList” auf Seite 140
- ältere Version eines Objekts wiederherstellen, siehe “restoreVersion” auf Seite 145

Referenzen prüfen

- Prüfung vor dem Löschen, ob die angegebenen WCM-Objekte von anderen WCM-Objekten referenziert werden, siehe “checkReferencesForDelete” auf Seite 123
- Prüfung vor der Freigabe, ob die angegebenen WCM-Objekte andere WCM-Objekte referenzieren, die noch nicht freigegeben wurden, siehe “checkReferencesForRelease” auf Seite 124

- Prüfung vor dem Vorlegen, ob die angegebenen WCM-Objekte andere WCM-Objekte referenzieren, die noch nicht zur Qualitätssicherung vorgelegt wurden, siehe “checkReferencesForSubmit” auf Seite 125
- Abfragen externer Referenzen, siehe “getExternalLinks” auf Seite 138

Deployment

- Seite(n) erzeugen, siehe “generatePage” auf Seite 132
- generierte Seite(n) aus dem Dateisystem löschen, siehe “depublishPage” auf Seite 130
- Abfragen verbleibender Deploymentaufgaben für ein angegebenes Deploymentsystem und ein WCM-Objekt, siehe “getDeploymentJobs” auf Seite 137

In der folgenden Übersicht sind die Methoden alphabetisch sortiert.

addRemark

Fügt eine Bemerkung zum Protokoll des spezifizierten WCM-Objekts hinzu

Parameter

Tabelle 47 – Parameter der Methode `addRemark`

Parameter	Datentyp	Beschreibung
`oid`	`ObjectId`	OID des betreffenden WCM-Objekts
`remark`	`String`	Zeichenkette, die zum Protokoll des WCM-Objekts hinzugefügt werden soll, oder `null`

Rückgabe

keine

Voraussetzungen

- Erforderliche Zugriffsrechte: `READ` und `WRITE_META`
- Objektstatus: keine Einschränkungen

Folge

Der Objektstatus ändert sich nicht.

change

Ändert die Metadaten eines WCM-Objekts

Es können nur die in Tabelle 13 “Die Komponenten des Datentyps `ObjectData`” auf Seite 79 als modifizierbar angegebenen Attribute geändert werden.

In `objectData.attributeKeys` müssen alle Attribute angegeben werden, die geändert werden sollen. Darüber hinaus muss `objectData.objectId` gesetzt sein.

Parameter

Tabelle 48 – Parameter der Methode `change`

Parameter	Datentyp	Beschreibung
`objectData`	`ObjectData`	Geänderte Objektdaten
`remark`	`String`	Zeichenkette, die zum Protokoll des WCM-Objekts hinzugefügt wird, oder `null`
`dplWaitInfo`	`DeploymentWaitInfo`	Liste der Deploymentsysteme oder `null`

Rückgabe

keine

Voraussetzungen

- Erforderliche Zugriffsrechte: `READ` und `WRITE_META`. Soll die Zugriffssteuerungsliste (ACL) geändert werden, ist zusätzlich das Zugriffsrecht `CHANGE_RIGHTS` notwendig.
- Objektstatus: muss `EDITED`, `REJECTED`, `RELEASED` oder `PENDING_RELEASE` sein

Folge

Nach der Änderung hat das angegebene WCM-Objekt den Status `EDITED`.

`checkReferencesForDelete`

Methode zur Prüfung, ob die angegebenen WCM-Objekte von anderen WCM-Objekten referenziert werden. Diese Methode sollte vor dem Löschen oder Zerstören von WCM-Objekten aufgerufen werden.

Parameter

Tabelle 49 – Parameter der Methode `checkReferencesForDelete`

Parameter	Datentyp	Beschreibung
`oids`	`ObjectId[]`	Liste von OIDs, deren zugehörige WCM-Objekte geprüft werden sollen

Rückgabe

`ObjectId[] objectIds`: eine Liste von OIDs, die in `oids` enthalten sind und deren zugehörige WCM-Objekte von anderen WCM-Objekten referenziert werden

Voraussetzungen

- Es sind keine Zugriffsrechte erforderlich
- Objektstatus: keine Einschränkungen

Folge

Der Objektstatus ändert sich nicht.

checkReferencesForRelease

Methode zur Prüfung, ob die angegebenen WCM-Objekte andere WCM-Objekte referenzieren, die noch nicht freigegeben wurden. Diese Methode sollte vor dem Freigeben von WCM-Objekten aufgerufen werden.

Parameter

Tabelle 50 – Parameter der Methode `checkReferencesForRelease`

Parameter	Datentyp	Beschreibung
`oids`	`ObjectId[]`	Liste von OIDs, deren zugehörige WCM-Objekte geprüft werden sollen

Rückgabe

`ObjectId[] objectIds`: Eine Liste von OIDs, die in `oids` enthalten sind und deren zugehörige WCM-Objekte auf andere WCM-Objekte referenzieren, die nicht in `oids` enthalten sind und noch nicht freigegeben wurden. Es werden nur solche WCM-Objekte zurückgegeben, die in der QS-Sicht verfügbar sind und somit mindestens einmal der Qualitätssicherung vorgelegt worden sind.

Voraussetzungen

- Es sind keine Zugriffsrechte erforderlich.
- Objektstatus: keine Einschränkungen

Folge

Der Objektstatus ändert sich nicht.

checkReferencesForSubmit

Methode zur Prüfung, ob die angegebenen WCM-Objekte andere WCM-Objekte referenzieren, die noch nicht zur Qualitätssicherung vorgelegt wurden. Diese Methode sollte vor dem Vorlegen von WCM-Objekten zur Qualitätssicherung aufgerufen werden.

Parameter

Tabelle 51 – Parameter der Methode `checkReferencesForSubmit`

Parameter	Datentyp	Beschreibung
`oids`	`ObjectId[]`	Liste von OIDs, deren zugehörige WCM-Objekte geprüft werden sollen

Rückgabe

`ObjectId[] objectIds`: eine Liste von OIDs, die in `oids` enthalten sind und deren zugehörige WCM-Objekte auf andere WCM-Objekte referenzieren, die nicht in `oids` enthalten sind und noch nicht zur Qualitätssicherung vorgelegt wurden

Voraussetzungen

- Es sind keine Zugriffsrechte erforderlich.
- Objektstatus: keine Einschränkungen

Folge

Der Objektstatus ändert sich nicht.

convertContent

Konvertiert den Inhalt eines Verbundobjekts. Dabei werden die untergeordneten Objekte eines Verbundobjekts angelegt.

Parameter

Tabelle 52 – Parameter der Methode convertContent

Parameter	Datentyp	Beschreibung
oid	ObjectId	OID des Verbundobjekts
dplWaitInfo	DeploymentWaitInfo	Liste der Deploymentsysteme oder null

Rückgabe

Keine

Voraussetzungen

- Erforderliches Zugriffsrecht auf das Quellobjekt: READ
- Erforderliche Zugriffsrechte auf das Thema in dem sich das Quellobjekt befindet: READ, CREATE, DELETE, WRITE und WRITE_META
- Objektstatus des Quellobjekts: muss EDITED, REJECTED, RELEASED oder PENDING_RELEASE sein

Folge

Der Objektstatus des Verbundobjekts ändert sich nicht. Der Objektstatus der untergeordneten Objekte, die während der Konvertierung erzeugt wurden, ist EDITED.

copy

Kopiert ein WCM-Objekt unter ein Thema

Das angegebene Objekt darf auch unter das Thema kopiert werden, in dem es sich bereits befindet. Dabei ändert sich der Titel des Objekts nicht. Es existieren dann zwei Objekte mit gleichem Namen.

Parameter

Tabelle 53 – Parameter der Methode copy

Parameter	Datentyp	Beschreibung
oid	ObjectId	OID des betreffenden WCM-Objekts
newParentOID	ObjectId	OID des Themas, unter das das WCM-Objekt kopiert werden soll
remark	String	Zeichenkette, die zum Protokoll des WCM-Objekts hinzugefügt wird, oder null
dplWaitInfo	DeploymentWaitInfo	Liste der Deploymentsysteme oder null

Rückgabe

ObjectId objectId: die OID des neuen, kopierten WCM-Objekts

Voraussetzungen

- Erforderliche Zugriffsrechte auf das kopierte WCM-Objekt: READ, TREE_OPERATIONS und WRITE_META
- Erforderliche Zugriffsrechte auf das Zielthema: READ und CREATE
- Objektstatus: muss EDITED, REJECTED, RELEASED oder PENDING_RELEASE sein

Folge

Der Status des kopierten WCM-Objekts (des Quellobjekts) ändert sich nicht. Die neu angelegte Kopie erhält den Status `EDITED`.

`create`

Erzeugt ein neues WCM-Objekt mit den angegebenen Metadaten. Die Vorlage in den Objektdaten (`objectData.template`) darf `null` sein.

Parameter

Tabelle 54 – Parameter der Methode `create`

Parameter	Datentyp	Beschreibung
`objectData`	`ObjectData`	Für ein neues Objekt müssen mindestens die Attribute: `topic`, `objectType` und `title` angegeben werden. Soll das neue Objekt auf einer Vorlage basieren so ist auch `template` anzugeben. Zugriffsrechte (ACL) können hier noch nicht definiert werden.
`content`	`byte[]`	Inhalt des Objekts oder `null`, falls kein Inhalt angegeben werden soll
`remark`	`String`	Zeichenkette, die zum Protokoll des WCM-Objekts hinzugefügt wird, oder `null`
`dplWaitInfo`	`DeploymentWaitInfo`	Liste der Deploymentsysteme oder `null`

Rückgabe

`ObjectId objectId`: die OID des neu erzeugten WCM-Objekts

Voraussetzungen

- Erforderliche Zugriffsrechte auf das übergeordnete Thema: `READ` und `CREATE`
- Objektstatus: keine Einschränkungen
- Die benötigten Funktionsbereiche hängen von dem zu erzeugenden Objekttyp ab.

Folge

Nach dem Anlegen hat das WCM-Objekt den Status `EDITED`.

`delete`

Löscht die angegebenen WCM-Objekte

Wurde ein WCM-Objekt noch nie zur Qualitätssicherung vorgelegt, so wird es durch diese Methode vollständig aus dem WCM-System entfernt. Anderenfalls erhält es den Objektstatus `DELETED` und bleibt weiterhin in der QS-Sicht sichtbar. Dort muss das Löschen mit einer `destroy`-Aktion bestätigt oder mittels `reject` abgelehnt werden.

Parameter

Tabelle 55 – Parameter der Methode `delete`

Parameter	Datentyp	Beschreibung
`oids`	`ObjectId[]`	Liste von OIDs
`remark`	`String`	Zeichenkette, die zum Protokoll der WCM-Objekte hinzugefügt wird, oder `null`
`emailInfo`	`EMailInfo`	E-Mail-Information, die an die in den Objektdaten definierten Empfänger gesendet wird, oder `null`, falls keine E-Mail gesendet werden soll

Parameter	Datentyp	Beschreibung
dplWaitInfo	DeploymentWaitInfo	Liste der Deploymentsysteme oder null

Rückgabe

keine

Voraussetzungen

- Erforderliche Zugriffsrechte: READ und DELETE
- Objektstatus: muss EDITED, REJECTED, RELEASED oder PENDING_RELEASE sein

Folge

Ist ein WCM-Objekt bereits einmal zur Qualitätssicherung vorgelegt worden, so erhält es nach dem Löschen den Status DELETED.

depublishPage

Löscht die für die angegebenen WCM-Objekte erzeugten Seiten aus dem Dateisystem. Die Seiten werden damit aus der Produktionssicht entfernt. Die gelöschten Seiten werden erst nach einer erneuten Freigabe der entsprechenden WCM-Objekte wieder erzeugt.

Parameter

Tabelle 56 – Parameter der Methode depublishPage

Parameter	Datentyp	Beschreibung
dplSystem	String	Name eines Deploymentsystems
oids	ObjectId[]	Liste von OIDs
dplWaitInfo	DeploymentWaitInfo	Liste der Deploymentsysteme oder null

Rückgabe

keine

Voraussetzungen

- Erforderliches Zugriffsrecht: `READ` und `RELEASE`
- Objektstatus: keine Einschränkungen

Folge

Der Objektstatus ändert sich nicht.

`destroy`

Zerstört die angegebenen WCM-Objekte und bestätigt damit das Löschen durch `delete`

Parameter

Tabelle 57 – Parameter der Methode `destroy`

Parameter	Datentyp	Beschreibung
`oids`	`ObjectId[]`	Liste von OIDs
`remark`	`String`	Zeichenkette, die zum Protokoll der WCM-Objekte hinzugefügt wird, oder `null`
`emailInfo`	`EMailInfo`	E-Mail-Information, die an die in den Objektdaten definierten Empfänger gesendet wird, oder `null`, falls keine E-Mail gesendet werden soll
`dplWaitInfo`	`DeploymentWaitInfo`	Liste der Deploymentsysteme oder `null`

Rückgabe

keine

Voraussetzungen

- Erforderliche Zugriffsrechte: `READ` und `RELEASE`
- Objektstatus: muss `DELETED` sein

Folge

Nach dem Zerstören ist das WCM-Objekt einschließlich aller alten Versionen nicht mehr im WCM-System sichtbar.

generatePage

Erzeugt die Seiten zu der angegebenen Liste von WCM-Objekten (Seitengenerierung). WCM-Objekte, deren generierte Seiten mithilfe der Funktion *Seite entfernen* aus der Produktionssicht entfernt wurden, können mit dieser Methode nicht neu generiert werden. Solche Objekte müssen noch einmal sämtliche Staging-Stufen durchlaufen, ehe sie wieder veröffentlicht werden können.

Parameter

Tabelle 58 – Parameter der Methode `generatePage`

Parameter	Datentyp	Beschreibung
`dplSystem`	`String`	Name eines Deploymentsystems
`oids`	`ObjectId[]`	Liste von OIDs
`useReleasedTemplate`	`boolean`	`true`, falls die freigegebene Vorlage verwendet werden soll, `false` für die aktuelle Version der Vorlage
`dplWaitInfo`	`DeploymentWaitInfo`	Liste der Deploymentsysteme oder `null`

Rückgabe

keine

Voraussetzungen

- Erforderliches Zugriffsrecht: `READ`
- Objektstatus: keine Einschränkungen

Folge

Der Objektstatus ändert sich nicht.

`get`

Liefert das WCM-Objekt mit der angegebenen OID in der definierten Version. Über den Parameter `attrKeys` kann eine genaue Liste von Attributen angefordert werden, die nach dem Aufruf der Methode im Rückgabewert verfügbar sind.

Parameter

Tabelle 59 – Parameter der Methode `get`

Parameter	Datentyp	Beschreibung
`oid`	`ObjectId`	OID des WCM-Objekts
`version`	`Version`	Gewünschte Version Die Zahl für die aktuelle Version kann hier nicht angegeben werden, `null` liefert die aktuelle Version.
`attributeKeys`	`String[]`	Liste der gewünschten Attribute

Rückgabe

`ObjectData objectData`: die angeforderten Daten des spezifizierten WCM-Objekts

Voraussetzungen

- Erforderliche Zugriffsrechte: `READ` bzw. `READ_PRODUCTION`
- Objektstatus: keine Einschränkungen

Folge

Der Objektstatus ändert sich nicht.

getCheckOutContent

Liefert den Inhalt des WCM-Objekts mit der angegebenen OID und der definierten Version in einem Byte-Array zurück. Diese Methode kann auch verwendet werden, wenn das WCM-Objekt nicht ausgeliehen ist, d.h. wenn der Status des WCM-Objekts nicht `CHECKED_OUT` ist.

Der Inhalt wird von dem Deploymentsystem generiert, das WCM WebServices zugeordnet ist. Bei HTML-Objekten und ähnlichen Objekttypen werden die Head-Bereiche der Vorlagen zusammengeführt und in den Head-Bereich des Inhalts eingefügt. Der Body-Bereich wird nicht geändert – bis auf die enthaltenen Referenzen, die ggf. auf den neuesten Stand gebracht werden. WCM-Tags werden nicht ersetzt. Die Head-Bereiche werden bei Themen nicht zusammengeführt.

Bei binären WCM-Objekten (wie z.B. GIF-Bildern) wird der Inhalt nicht geändert.

Parameter

Tabelle 60 – Parameter der Methode `getCheckOutContent`

Parameter	Datentyp	Beschreibung
`oid`	`ObjectId`	OID des WCM-Objekts
`version`	`Version`	Gewünschte Version, `null` liefert die aktuelle Version

Rückgabe

`byte[] content`: der ausgeliehene Inhalt des angeforderten WCM-Objekts

Voraussetzungen

- Erforderliches Zugriffsrecht: `READ`
- Objektstatus: keine Einschränkungen

Folge

Der Objektstatus ändert sich nicht.

`getChildren`

Liefert die OIDs der Kind-Objekte des angegebenen WCM-Objekts unter der spezifizierten Ansicht `view`

Parameter

Tabelle 61 – Parameter der Methode `getChildren`

Parameter	Datentyp	Beschreibung
`oid`	`ObjectId`	OID des WCM-Objekts
`view`	`int`	`1` = Themenbaumansicht `2` = Vorlagenansicht `3` = Verbundobjektansicht

Rückgabe

`ObjectId[] objectIds`: eine Liste von OIDs der angeforderten Kind-Objekte

Voraussetzungen

- Erforderliches Zugriffsrecht: `READ`
- Objektstatus: keine Einschränkungen

Folge

Der Objektstatus ändert sich nicht.

getContent

Liefert den Inhalt des WCM-Objekts mit der angegebenen OID und der definierten Version in einem Byte-Array zurück. Diese Methode liefert den Inhalt des WCM-Objekts, wie er im Content-Repository gespeichert ist, ohne weitere Manipulationen am Inhalt in Bezug auf die verwendete Vorlage durchzuführen.

> **Hinweis:** Wenn das WCM-Objekt keinen Inhalt hat, wird beim Aufruf dieser Methode eine Exception geworfen.

Parameter

Tabelle 62 – Parameter der Methode `getContent`

Parameter	Datentyp	Beschreibung
`oid`	`ObjectId`	OID des WCM-Objekts
`version`	`Version`	Gewünschte Version, `null` liefert den Inhalt der aktuellen Version

Rückgabe

`byte[] content`: der Inhalt des angeforderten WCM-Objekts

Voraussetzungen

- Erforderliches Zugriffsrecht: `READ`
- Objektstatus: keine Einschränkungen

Folge

Der Objektstatus ändert sich nicht.

getDeploymentJobs

Liefert alle verbleibenden Deploymentaufgaben für ein angegebenes Deploymentsystem und ein WCM-Objekt zurück. Jede Staging-Aktion, die ein WCM-Objekt ändert, erzeugt Deploymentaufgaben für das Deploymentsystem einer Website.

Parameter

Tabelle 63 – Parameter der Methode `getDeploymentJobs`

Parameter	Datentyp	Beschreibung
`dplSystem`	`String`	Name des Deploymentsystems
`oid`	`ObjectId`	OID des betreffenden WCM-Objekts

Rückgabe

`DeploymentJobInfo[] deplJobs`: Die Liste der verbleibenden Deploymentaufgaben. Gibt es keine Aufgaben für die angegebenen Parameter, so wird eine leere Liste zurückgeliefert.

Voraussetzungen

- Es sind keine Zugriffsrechte erforderlich.
- Objektstatus: keine Einschränkungen

Folge

Der Objektstatus ändert sich nicht.

getExternalLinks

Liefert alle externen Referenzen, die in der Liste der angegebenen WCM-Objekte enthalten sind. Externe Referenzen zeigen auf URLs außerhalb der WCM-verwalteten Website.

Parameter

Tabelle 64 – Parameter der Methode `getExternalLinks`

Parameter	Datentyp	Beschreibung
`oids`	`ObjectId[]`	OIDs der zu betrachtenden WCM-Objekte

Rückgabe

`LinkInfo[] links`: die Liste der externen Links

Voraussetzungen

- Erforderliches Zugriffsrecht: `READ`
- Objektstatus: keine Einschränkungen

Folge

Der Objektstatus ändert sich nicht.

getLastLogEntries

Liefert eine Liste von Protokolleinträgen zum angegebenen WCM-Objekt. Die Protokolleinträge werden in absteigender Reihenfolge geliefert, beginnend bei `first`.

Parameter

Tabelle 65 – Parameter der Methode `getLastLogEntries`

Parameter	Datentyp	Beschreibung
`oid`	`ObjectId`	OID des WCM-Objekts
`first`	`int`	Nummer des ersten zu liefernden Protokolleintrags; -1 bedeutet beginnend mit dem letzten Protokolleintrag
`number`	`int`	Anzahl der zu liefernden Protokolleinträge; -1 bedeutet alle Einträge bis zum ersten

Rückgabe

`LogEntry[] logEntries`: die Liste der angeforderten Protokolleinträge

Voraussetzungen

- Erforderliches Zugriffsrecht: `READ`
- Objektstatus: keine Einschränkungen

Folge

Der Objektstatus ändert sich nicht.

`getParent`

Liefert die OIDs der WCM-Objekte, die in der angegebenen Ansicht `view` dem spezifizierten WCM-Objekt übergeordnet sind (“Eltern-Objekte”)

Parameter

Tabelle 66 – Parameter der Methode `getParent`

Parameter	Datentyp	Beschreibung
`oid`	`ObjectId`	OID des WCM-Objekts
`view`	`int`	`1` = Themenbaumansicht `2` = Vorlagenansicht `3` = Verbundobjektansicht
`toRoot`	`boolean`	`true`, falls alle Eltern-Objekte bis zur Wurzel zurückgeliefert werden sollen

Rückgabe

`ObjectId[] objectIds`: Eine Liste von OIDs der angeforderten Eltern-Objekte. Hatte `toRoot` den Wert `false`, so wird genau ein Objekt zurückgeliefert. Für das Wurzel-Objekt wird eine leere Liste zurückgeliefert.

Voraussetzungen

- Es sind keine Zugriffsrechte erforderlich.
- Objektstatus: keine Einschränkungen

Folge

Der Objektstatus ändert sich nicht.

`getVersionList`

Liefert die vollständige Liste aller Versionen des angegebenen WCM-Objekts

Parameter

Tabelle 67 – Parameter der Methode `getVersionList`

Parameter	Datentyp	Beschreibung
`oid`	`ObjectId`	OID des betreffenden WCM-Objekts

Rückgabe

`Version[] versions`: die angeforderte Liste aller Versionen zum gegebenen WCM-Objekt

Voraussetzungen

- Erforderliches Zugriffsrecht: `READ`
- Objektstatus: keine Einschränkungen

Folge

Der Objektstatus ändert sich nicht.

`move`

Verschiebt das angegebene WCM-Objekt unter ein anderes Thema

Parameter

Tabelle 68 – Parameter der Methode `move`

Parameter	Datentyp	Beschreibung
`oid`	`ObjectId`	OID des betreffenden WCM-Objekts
`newParentOID`	`ObjectId`	OID des Themas, unter das das WCM-Objekt verschoben werden soll

Parameter	Datentyp	Beschreibung
remark	String	Zeichenkette, die zum Protokoll des WCM-Objekts hinzugefügt wird, oder null
dplWaitInfo	DeploymentWaitInfo	Liste der Deploymentsysteme oder null

Rückgabe

keine

Voraussetzungen

- Erforderliche Zugriffsrechte auf das verschobene WCM-Objekt: READ, TREE_OPERATIONS und WRITE_META
- Erforderliche Zugriffsrechte auf das Zielthema: READ und CREATE
- Objektstatus: muss EDITED, RELEASED oder PENDING_RELEASE sein

Folge

Nach dem Verschieben hat das angegebene WCM-Objekt den Status EDITED.

multiImport

Erzeugt mehrere neue WCM-Objekte in einem Aufruf. Hierzu wird das initiale Objekt (imports.objectData) angelegt und anschließend die in imports.children[] definierten Objekte. Der Import unterscheidet dabei die folgenden Möglichkeiten:

- Wenn das initiale Objekt einen Inhalt hat (imports.content != null), so wird dieses als Startseite des Imports aufgefasst und von den in imports.children[] angegebenen Objekten werden nur die direkt oder indirekt von diesem WCM-Objekt referenzierten Objekte angelegt.

- Hat das initiale Objekt keinen Inhalt, werden alle übergebenen Objekte angelegt.

Beim Anlegen der einzelnen Objekte werden explizite Hyperlinks in die bestehende Website und relative Hyperlinks auf mit importierte Objekte erkannt und in die Referenzenverwaltung von Livelink WCM Server übernommen.

Parameter

Tabelle 69 – Parameter der Methode `multiImport`

Parameter	Datentyp	Beschreibung
`initialData`	`ObjectData`	Hier werden das Thema (`topic`), der Objekttyp (`objectType`), der Vorschlag für den Dateinamen (`deploymentHint`) und der Titel (`title`) des neu anzulegenden Objekts angegeben. Alle anderen Attribute werden automatisch für die untergeordneten Objekte übernommen und überschreiben die in `imports.children[].objectData` angegebenen Werte. So ist es z. B. möglich, einheitliche Attribute für alle Objekte vorzugeben. Attribute, die nur für das Startobjekt des Imports gelten, sollten Sie in `imports.children[0].objectData` festlegen. Der Import erzeugt ein eigenes Thema, wenn `initialData.objectType` vom Typ Thema ist. Anderenfalls werden die anzulegenden Objekte unterhalb des Themas `initialData.topic` angelegt.

Parameter	Datentyp	Beschreibung
`imports`	`MultiImportPart`	Metadaten, Inhalt und untergeordnete Objekte Siehe “Datentyp: `MultiImportPart`” auf Seite 92
`remark`	`String`	Zeichenkette, die zum Protokoll sämtlicher erzeugter WCM-Objekte hinzugefügt wird, oder `null`
`dplWaitInf`	`DeploymentWaitInfo`	Liste der Deploymentsysteme oder `null`

Rückgabe

`ObjectId objectId`: die OID des Startobjekts für den Import

Voraussetzungen

- Erforderliche Zugriffsrechte auf das übergeordnete Thema: `READ` und `CREATE`
- Objektstatus: keine Einschränkungen
- Die benötigten Funktionsbereiche hängen von den zu erzeugenden Objekttypen ab.

Folge

Nach dem Import haben sämtliche neu angelegten WCM-Objekte den Status `EDITED`.

restoreVersion

Stellt eine frühere Version des angegebenen WCM-Objekts wieder her

Parameter

Tabelle 70 – Parameter der Methode `restoreVersion`

Parameter	Datentyp	Beschreibung
`oid`	`ObjectId`	OID des betreffenden WCM-Objekts
`version`	`Version`	Gewünschte Version des WCM-Objekts
`remark`	`String`	Zeichenkette, die zum Protokoll des WCM-Objekts hinzugefügt wird, oder `null`
`dplWaitInfo`	`DeploymentWaitInfo`	Liste der Deploymentsysteme oder `null`

Rückgabe

keine

Voraussetzungen

- Erforderliche Zugriffsrechte: `READ`, `WRITE` und `WRITE_META`
- Objektstatus: muss `EDITED`, `REJECTED`, `RELEASED` oder `PENDING_RELEASE` sein

Folge

Nach der Wiederherstellung der angegebenen Version hat das WCM-Objekt den Status `EDITED`.

sortParentsFirst

Liefert eine Liste zurück, in der jedes Kind-Objekt bezogen auf die spezifizierte Ansicht `view` hinter seinem direkten Eltern-Objekt platziert ist. Diese Methode ist nützlich, um WCM-Objekte zu sortieren, die zur Qualitätssicherung vorgelegt oder freigegeben werden sollen.

Parameter

Tabelle 71 – Parameter der Methode `SortParentsFirst`

Parameter	Datentyp	Beschreibung
`oids`	`ObjectId[]`	OIDs der zu sortierenden WCM-Objekte
`topologyView`	`int`	`1` = Themenbaumansicht `2` = Vorlagenansicht `3` = Verbundobjektansicht

Rückgabe

`ObjectId[] objectIds`: eine Liste, die die neu sortierten OIDs enthält

Voraussetzungen

- Es sind keine Zugriffsrechte erforderlich
- Objektstatus: keine Einschränkungen

Folge

Der Objektstatus ändert sich nicht.

5.6 Suchen nach WCM-Objekten

Die Suche nach WCM-Objekten wird durch die Methode `filter` ermöglicht. Eine Suchbedingung wird dabei als ein Objekt vom Typ `Filter` repräsentiert. Mit den auf dem Typ `Filter` basierenden Datentypen lassen sich logische Ausrücke modellieren. Dabei können mithilfe von Verknüpfungen komplexe Filter zusammengestellt werden.

Es gibt verschiedene Kategorien von Filtertypen:

- Filter, die auf den Attributen der WCM-Objekte basieren (Informationen zu suchbaren Attributen enthält Tabelle 13 “Die Komponenten des Datentyps `ObjectData`” auf Seite 79). Diese Filter sind vom Typ `IsNullFilter`, `JoinFilter` oder erweitern den Datentyp `ValueFilter`.
- Filter, die schon vorhandene Filter logisch miteinander verknüpfen und dadurch die Konstruktion von komplexen Suchanfragen ermöglichen. Folgende Filtertypen gehören zu dieser Kategorie: `NotFilter`, `AndFilter` und `OrFilter`.
- Filter, die eine vordefinierte Suchfunktion ausführen. Folgende Filtertypen gehören zu dieser Kategorie: `RootTemplateFilter`, `PermissionFilter`, `SubtreeFilter` und `PrincipalFilter`.

Die folgende Übersicht veranschaulicht den Datentyp `Filter` und davon abgeleitete Filtertypen.

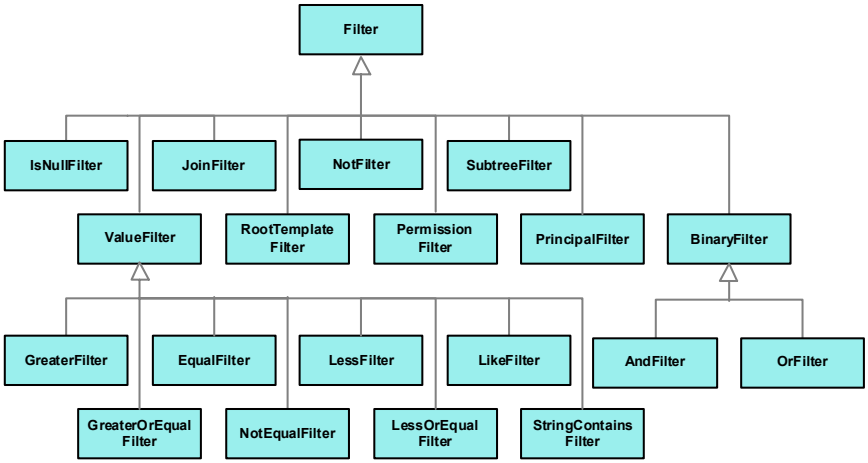


Abb. 7 – Der Datentyp `Filter` und davon abgeleitete spezielle Filtertypen

Attributwerte-Filter (`ValueFilter`)

Livelink WCM Server verwaltet die Daten einer Website (einschließlich der Inhalte und der Metadaten) in einer Datenbank. Bei der Suche wird auf Basis eines Filters eine für das jeweilige Datenbanksystem geeignete Suchanfrage (eine SQL-Anweisung der Form `SELECT ... FROM ... WHERE ... ORDER BY ...`) formuliert. Die Attributnamen entsprechen den Komponentennamen des Datentyps `ObjectData`.

Es gibt jedoch einige Einschränkungen bezüglich der Verwendung von Attributen für die Formulierung von SQL-Anfragen. Aus diesem Grund können nicht alle Attribute in einer Suche verwendet werden. Genaue Angaben zu den suchbaren Attributen finden Sie in Tabelle 13 “Die Komponenten des Datentyps `ObjectData`” auf Seite 79.

Anhand der folgenden Tabelle kann abgelesen werden, welche Attributtypen mit welchen Filtertypen zusammen verwendet werden können.

Tabelle 72 – Attributtypen und Filter

	LikeFilter	StringContainsFilter	IsNullFilter[a]	JoinFilter[b]	EqualFilter	NotEqualFilter	GreaterFilter	GreaterOrEqualFilter	LessFilter	LessOrEqualFilter
IntegerValue, LongValue			✓	✓	✓	✓	✓	✓	✓	✓
ListValue[c], SetValue[c]	✓	✓	✓							
DateValue			✓	✓	✓	✓	✓	✓	✓	✓
BooleanValue			✓	✓	✓	✓				
StringValue	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
LocaleValue[d], ObjectId, ObjectType, ObjectState, User[d], Version				✓ ✓	✓					

(a) Mit diesem Filter kann getestet werden, ob der Wert für ein Attribut gesetzt wurde.

(b) Dieser Filter vergleicht zwei Attributwerte miteinander. Beide Werte müssen vom gleichen Typ sein.

(c) Gesucht wird jeweils nach einem Eintrag aus der Liste, der als `StringValue` angegeben wird.

(d) Bei `LocaleValue` und `User` kann der Wert auch als `StringValue` angegeben werden, dadurch können auch die anderen Filter verwendet werden.

Beispiele

```
new IsNullFilter("subtitle")

new StringContainsFilter("QAEMailReceivers",
      new StringValue("jstein@company.example"))

new LikeFilter("QAEMail*",
      new StringValue("jstein@*"))

new EqualFilter("createdBy", new StringValue("jstein"))
```

Vordefinierte Suchfunktionen als Filter

- Der Filtertyp `RootTemplateFilter` sucht alle WCM-Objekte einer Website, die vom Objekttyp “Vorlage” sind und die selbst keine Vorlage besitzen.
- Der Filtertyp `PermissionFilter` sucht nach allen WCM-Objekten, bei denen ein vorgegebener Benutzer mit einem vorgegebenen Recht in die Zugriffssteuerungsliste eingetragen wurde.
- Mit dem Filtertyp `SubtreeFilter` können alle WCM-Objekte unterhalb eines gegebenen WCM-Objekts (Thema in der Themenstruktur bzw. Vorlage in der Vorlagenstruktur) ermittelt werden. Der `SubtreeFilter` kann auch ohne Angabe des `Topology`-Parameters verwendet werden. In diesem Fall gilt die Themenstruktur als Defaultwert.
- Der Filtertyp `PrincipalFilter` sucht nach WCM-Objekten, denen ein bestimmter Principal (Benutzer, Gruppe oder Rolle) zugeordnet ist. Zusätzlich besteht die Möglichkeit, das Ergebnis auf WCM-Objekte einzuschränken, für die der Principal ein bestimmtes Recht (`Permission`) besitzt. Weitere Informationen zu Principals finden Sie u.a. in Abschnitt 4.1 “Benutzer, Gruppen und Rollen” auf Seite 54.

Beispiele

Der folgende Filter sucht alle WCM-Objekte, die der Benutzer “jstein” lesen darf.

```
new PermissionFilter(new Permission("r"), new User("jstein") )
```

Mit dem folgenden Filter werden alle WCM-Objekte (einschließlich des spezifizierten Startknotens) zurückgegeben, die in der Themenstruktur unter dem WCM-Objekt mit der OID “4711” liegen.

```
new SubtreeFilter(new ObjectId("4711") )
```

Die Methode `filter`

Diese Methode findet Objekte, die dem definierten Filterkriterium entsprechen, unter dem angegebenen Thema.

Das Ergebnis kann sortiert werden durch Spezifizieren einer Liste von Sortierobjekten, die die zu sortierenden Attribute und die Sortierreihenfolge festlegt. Es ist außerdem möglich, Spezialattribute zur Konstruktion eines Suchfilters zu verwenden.

> **Hinweis:** Bei der Angabe von Spezialattributen für die Sortierung der Ergebnisliste arbeitet die `filter`-Methode zurzeit noch nicht einwandfrei.

WCM-Objekte, für die der Benutzer der Methode keine Leserechte hat, werden automatisch von der Suche ausgeschlossen.

Das WCM-Objekt, das als Startpunkt der Suche definiert wird, muss ein Thema sein.

Die Ergebnismenge kann durch `startResult` und `numberOfResults` begrenzt werden.

Parameter

Tabelle 73 – Parameter der Methode `filter`

Parameter	Datentyp	Beschreibung
`filter`	`Filter`	Suchkriterium oder `null`, um nach allen Objekten zu suchen
`startOID`	`ObjectId`	Durch die Angabe der OID eines Startknotens erfolgt die Suche erst unterhalb des angegebenen WCM-Objekts. In diesem Fall werden der Startknoten `startOID` selbst und alle in der Themenstruktur untergeordneten WCM-Objekte berücksichtigt. Die `startOID` muss ein WCM-Objekt vom Typ “Thema” referenzieren.
`sortList`	`Sort[]`	Liste von Attributkonstanten, nach denen die Ergebnismenge sortiert werden soll. Standardmäßig werden die gefundenen WCM-Objekte nach der OID sortiert.
`startResult`	`int`	Zahl, die das erste Element des Suchergebnisses angibt (meistens 0)
`numberOfResults`	`int`	Anzahl der zurückzuliefernden Suchergebnisse, beginnend bei `startResult`. -1 für alle Ergebnisse
`attributeKeys`	`String[]`	Namen aller Attribute, die in den zurückgelieferten Objektdaten gesetzt sein sollen

Rückgabe

`ObjectData[] objectData`: Eine Liste von Objektdaten. Dies sind die Daten der WCM-Objekte, auf die das angegebene Suchkriterium passt.

Voraussetzungen

- Erforderliches Zugriffsrecht: `READ`
- Objektstatus: keine Einschränkungen

Folge

Der Objektstatus ändert sich nicht.

KAPITEL 6

Anwendungsbeispiele

Ein zentrales Anliegen der Web-Service-Technologie ist das einfache Auffinden und die einfache Verwendung eines Web-Service. Der erste Punkt wird von UDDI übernommen und ist nicht Thema dieses Handbuchs. Da sich Web-Services durch WSDL selbst beschreiben, gibt es von vielen Herstellern Werkzeuge, die eine Web-Service-Definition einlesen und automatisch in ein für eine Client-Anwendung nutzbares Modul umwandeln. Als Anwendungsentwickler binden Sie dieses Modul in Ihre Entwicklungsumgebung ein und schreiben Ihre Anwendung in der von dem Werkzeug unterstützten Programmiersprache.

In diesem Kapitel wird an zwei Beispielen gezeigt, wie WCM WebServices genutzt werden kann. Aus der großen Vielfalt von Programmiersprachen, die Web-Services nutzen können, werden hier VisualBasic for Applications (VBA) stellvertretend für nicht objekt-orientierte Skriptsprachen und C# im Zusammenspiel mit VisualStudio.NET für die Beispiele benutzt.

Es ist nicht Ziel dieses Kapitels, einen Programmierkurs für diese Sprachen zu bieten, sondern vielmehr auf die Besonderheiten im Zusammenhang mit ihrer Nutzung hinzuweisen und prototypisch lauffähigen Code zu präsentieren.

Die Beispiele beziehen sich auf die fiktive Website “InternetSite” der Firma “company.example”:

- Beispiel 1 demonstriert, wie komplexe Filterausdrücke formuliert und die Ergebnismengen bearbeitet werden.
- Beispiel 2 zeigt, wie die Staging-Funktionen von Livelink WCM Server genutzt werden.

Im Rahmen dieser Beispiele lernen Sie auch, wie in den beiden Programmierumgebungen die Authentifizierung, das Session-Management und die Fehlerbehandlung gehandhabt werden.

6.1 VisualBasic for Applications

VisualBasic for Applications wird immer dann eingesetzt, wenn neue Funktionen in Microsoft Office-Anwendungen integriert werden sollen. Zur Nutzung von Web-Services stellt Microsoft das so genannte “SOAP-Toolkit” zur Verfügung. Dieser muss auf den Client-Rechnern installiert werden. Das SOAP-Toolkit ist für den korrekten Versand und Empfang der SOAP-Nachrichten zuständig, den XML-Inhalt müssen Sie als Anwendungsentwickler generieren. Daher lassen sich die in diesem Kapitel beschriebenen Techniken auch auf andere Skriptsprachen übertragen.

Aufbau eines Web-Service-Aufrufs

Zur Vorbereitung soll zunächst der Aufbau eines Web-Service-Aufrufs erläutert werden:

Jeder Aufruf beginnt jeweils mit dem SOAP-Envelope, gefolgt vom SOAP-Body. Dieser enthält genau ein Element, repräsentiert durch den Namen der aufzurufenden Methode. Dieses Element enthält wiederum die Parameter der Methode. Leere Parameter dürfen dabei weggelassen werden.

Das folgende Beispiel zeigt einen vollständigen Request (inklusive HTTP-Header), wie er im .NET Framework SDK von Microsoft für den Methodenaufruf

```
filter('objectType == PDF', ...)
```

erzeugt wird:

```
POST /wcm/WebService/Port/InternetSite/edit HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client
Protocol 1.0.3705.0)
Content-Type: text/xml;
charset=utf-8
SOAPAction: "http://gaussvip.com/"
Authorization: Basic YWRtaW46YQ==
Content-Length: 1820
Expect: 100-continue
Host: wcmserver.company.example
Cookie: JSESSIONID=B8D157BC6E214885EE2A2A3A0FA788C8

<?xml version="1.0" encoding="utf-8"?>
 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
              xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
                xmlns:tns="http://gaussvip.com/"
                xmlns:types="http://gaussvip.com/encodedTypes"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body
         soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <tns:filter>
    <filter href="#id1" />
    <startOID href="#id2" />
    <sortList href="#id3" />
    <startResult xsi:type="xsd:int">0</startResult>
    <numberOfResults xsi:type="xsd:int">5</numberOfResults>
    <attributeKeys href="#id4" />
   </tns:filter>
   <tns:EqualFilter id="id1" xsi:type="tns:EqualFilter">
    <key href="#id5" />
    <value href="#id6" />
    <caseSensitive xsi:type="xsd:boolean">false</caseSensitive>
   </tns:EqualFilter>
   <tns:ObjectId id="id2" xsi:type="tns:ObjectId">
    <id xsi:type="xsd:string">1 </id>
   </tns:ObjectId>
   <soapenc:Array id="id3" soapenc:arrayType="tns:Sort[1]">
    <Item href="#id7" />
   </soapenc:Array>
   <soapenc:Array id="id4" soapenc:arrayType="xsd:string[4]">
    <Item>createdDate</Item>
    <Item>objectId   </Item>
    <Item>objectState</Item>
    <Item>title      </Item>
   </soapenc:Array>
   <tns:Key id="id5" xsi:type="tns:Key">
```

```
    <stringValue xsi:type="xsd:string">objectType</stringValue>
   </tns:Key>
   <tns:ObjectType id="id6" xsi:type="tns:ObjectType">
    <name xsi:type="xsd:string">PDF</name>
    <template xsi:type="xsd:boolean">false</template>
    <topic xsi:type="xsd:boolean">false   </topic>
    <frame xsi:type="xsd:boolean">false   </frame>
    <fileOnCreateNeeded xsi:type="xsd:boolean">false
    </fileOnCreateNeeded>
   </tns:ObjectType>
   <tns:Sort id="id7" xsi:type="tns:Sort">
    <stringValue xsi:type="xsd:string">createdDate</stringValue>
    <descending xsi:type="xsd:boolean">false      </descending>
   </tns:Sort>
  </soap:Body>
 </soap:Envelope>
```

Dieses Beispiel ist sehr komplex. Die Ursache hierfür liegt in dem allgemeinen Lösungsansatz, den das .NET Framework SDK (wie andere Toolkits auch) für die Serialisierung von Objekten in XML wählt: Es wird versucht, gleiche Objekte nur genau einmal zu serialisieren. Das ist nicht zwingend erforderlich. Deshalb kann ein äquivalenter SOAP-Befehl auch so einfach aussehen:

```
POST /wcm/WebService/Port/InternetSite/edit HTTP/1.1
Authorization: Basic YWRtaW46YQ==
Content-Type: text/xml
Host: wcmserver.company.example
SOAPAction: "http://gaussvip.com/
"Content-Length: 1378

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
 <SOAP-ENV:Envelope
       xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
       xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       xmlns:vip="http://gaussvip.com/">
  <SOAP-ENV:Body>
   <SOAPSDK1:filter xmlns:SOAPSDK1="http://gaussvip.com/">
    <filter xsi:type="vip:EqualFilter">
     <key xsi:type="vip:Key">
      <stringValue xsi:type="xsd:string">objectType</stringValue>
     </key>
```

```
     <value xsi:type="vip:ObjectType">
      <name type="xsd:string">PDF</name>
     </value>
    </filter>
    <startOID xsi:type="vip:ObjectId">
     <id xsi:type="xsd:string">1</id>
    </startOID>
    <sortList
           xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="vip:Sort[1]">
     <Item xsi:type="vip:Sort">
      <stringValue xsi:type="xsd:string"> createdDate</stringValue>
      <descending  xsi:type="xsd:boolean">False      </descending>
     </Item>
    </sortList>
    <startResult     xsi:type="xsd:int">0</startResult>
    <numberOfResults xsi:type="xsd:int">5</numberOfResults>
    <attributeKeys
          xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="xsd:string[4]">
     <Item xsi:type="xsd:string">createdDate  </Item>
     <Item xsi:type="xsd:string">objectId     </Item>
     <Item xsi:type="xsd:string">objectState  </Item>
     <Item xsi:type="xsd:string">title        </Item>
    </attributeKeys>
   </SOAPSDK1:filter>
  </SOAP-ENV:Body>
 </SOAP-ENV:Envelope>
```

In den folgenden Abschnitten werden wir die Konstruktion solcher einfach formulierten SOAP-Befehle demonstrieren. Die vom Server ebenfalls als SOAP-Envelope empfangene Antwort ist mindestens genauso komplex wie die oben abgedruckte Anfrage. Hier lässt sich die Komplexität leider nicht steuern. Der aufwendigere Teil des Programmierbeispiels behandelt deshalb die Deserialisierung der empfangenen Daten.

```
HTTP/1.1 100 Continue
HTTP/1.1 200 OK
Content-Type: text/xml;
charset=utf-8
Content-Length: 4151
Date: Thu, 18 Apr 2002 07:30:32 GMT
Server: Apache Tomcat/4.0.3 (HTTP/1.1 Connector)

<?xml version="1.0" encoding="UTF-8"?>
```

```
<SOAP-ENV:Envelope
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
         xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <SOAP-ENV:Body>
   <ns1:filterResponse
   SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
         xmlns:ns1="http://gaussvip.com/">
   <filterResult xsi:type="SOAP-ENC:Array"
         SOAP-ENC:arrayType="ns1:ObjectData[5]"
         xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
    <item href="#id1"/>
    <item href="#id2"/>
    <item href="#id3"/>
    <item href="#id4"/>
    <item href="#id5"/>
   </filterResult>
   </ns1:filterResponse>
   <multiRef id="id5" SOAP-ENC:root="0" xsi:type="ns3:ObjectData"
        xmlns:ns3="http://gaussvip.com/"
        xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">

   <createdDate xsi:type="xsd:dateTime">2002-04-17T16:45:32.000Z
   </createdDate>
   <objectId href="#id6"/>
   <objectState href="#id7"/>
   <title xsi:type="xsd:string">8.0.1_portalManagerProgrammersManual_en
   </title>
  </multiRef>
  <multiRef id="id2" SOAP-ENC:root="0" xsi:type="ns4:ObjectData"
            xmlns:ns4="http://gaussvip.com/"
            xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
   <createdDate xsi:type="xsd:dateTime">2002-04-17T16:45:08.000Z
   </createdDate>

   <objectId href="#id8"/>
   <objectState href="#id7"/>
   <title
         xsi:type="xsd:string">8.0.1_contentManagerProgrammersManual_en
   </title>
  </multiRef>
  <multiRef id="id4" SOAP-ENC:root="0" xsi:type="ns5:ObjectData"
            xmlns:ns5="http://gaussvip.com/"
            xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
   <createdDate xsi:type="xsd:dateTime">2002-04-17T16:45:24.000Z
   </createdDate>
   <objectId href="#id9"/>
   <objectState href="#id7"/>
```

```
 <title xsi:type="xsd:string">8.0.1_contentMinerManual_en
 </title>
</multiRef>
<multiRef id="id3" SOAP-ENC:root="0" xsi:type="ns6:ObjectData"
          xmlns:ns6="http://gaussvip.com/"
          xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
 <createdDate xsi:type="xsd:dateTime">2002-04-17T16:45:18.000Z
 </createdDate>
 <objectId href="#id10"/>
 <objectState href="#id7"/>
 <title xsi:type="xsd:string">8.0.1_contentManagerUserManual_en
 </title>
</multiRef>
<multiRef id="id1" SOAP-ENC:root="0" xsi:type="ns7:ObjectData"
          xmlns:ns7="http://gaussvip.com/"
          xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
 <createdDate xsi:type="xsd:dateTime">2002-04-17T16:45:06.000Z
 </createdDate>
 <objectId href="#id11"/>
 <objectState href="#id7"/>
 <title xsi:type="xsd:string">VipNote20_PomaMigration_en
 </title>
</multiRef>
<multiRef id="id6" SOAP-ENC:root="0" xsi:type="ns8:ObjectId"
          xmlns:ns8="http://gaussvip.com/"
          xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
 <id xsi:type="xsd:string">62</id>
</multiRef>

<multiRef id="id9" SOAP-ENC:root="0" xsi:type="ns9:ObjectId"
          xmlns:ns9="http://gaussvip.com/"
          xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
 <id xsi:type="xsd:string">61</id>
</multiRef>
<multiRef id="id10" SOAP-ENC:root="0" xsi:type="ns10:ObjectId"
          xmlns:ns10="http://gaussvip.com/"
          xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
 <id xsi:type="xsd:string">60</id>
</multiRef>
<multiRef id="id8" SOAP-ENC:root="0" xsi:type="ns11:ObjectId"
          xmlns:ns11="http://gaussvip.com/"
          xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
 <id xsi:type="xsd:string">59</id>
</multiRef>
<multiRef id="id7" SOAP-ENC:root="0" xsi:type="ns12:ObjectState"
          xmlns:ns12="http://gaussvip.com/"
          xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
 <name xsi:type="xsd:string">edited </name>
 <description xsi:type="xsd:string">ge&#228;ndert</description>
```

```
    <imageURL xsi:type="xsd:string">
       http://wcmserver.company.example/wcm/InternetSite_edit/
                                    vipimages/objectstate/changed.gif
    </imageURL>
   </multiRef>
   <multiRef id="id11" SOAP-ENC:root="0" xsi:type="ns13:ObjectId"
             xmlns:ns13="http://gaussvip.com/"
             xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
    <id xsi:type="xsd:string">58</id>
   </multiRef>
   </SOAP-ENV:Body>
 </SOAP-ENV:Envelope>
```

Vorarbeiten

Das SOAP-Toolkit steht unter http://download.microsoft.com/download/xml/soap/2.0/W98NT42KMe/EN-US/SoapToolkit20.exe zum Download zur Verfügung. Es beinhaltet die Version 3 (MSXML3) des Microsoft XML-Parsers. Wenn Sie beispielsweise ein Add-In für Microsoft Word erstellen wollen, legen Sie in Word eine Dokumentvorlage an und öffnen den VisualBasic-Editor mit ALT+ F11. Das SOAP-Toolkit und den XML-Parser müssen Sie als Verweise (im Menü *Extras* → *Verweise*) eintragen:

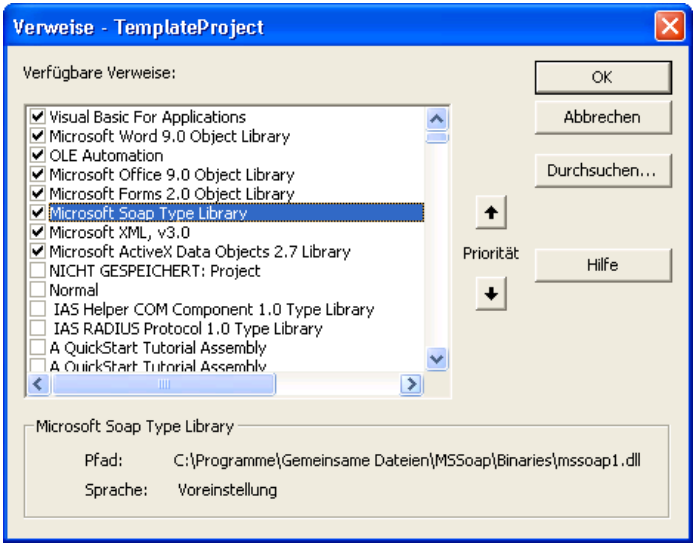


Abb. 8 – Einbindung der Bibliotheken im VisualBasic Editor

Architektur der VBA-Anwendung

Die folgenden Klassenmodule werden definiert:

Tabelle 74 – Klassenmodule

Name	Beschreibung	WCM-Objekttyp
`VipWebServiceClient`	Schnittstelle zur Nutzung von WCM WebServices. Neben der Initialisierung unter Angabe der Service-URL, der Benutzerkennung und des Passworts stellt dieses Modul alle in WCM WebServices definierten Methoden als Funktionen zur Verfügung.	

Name	Beschreibung	WCM-Objekttyp
`ObjectId`	Kapselt die OID eines WCM-Objekts Erlaubt Serialisierung und Deserialisierung	`ObjectId`
`ObjectState`	Kapselt den Status eines WCM-Objekts Erlaubt Serialisierung und Deserialisierung	`ObjectState`
`ObjectType`	Kapselt den Typ eines WCM-Objekts Erlaubt Serialisierung und Deserialisierung	`ObjectType`
`GenericFilter`	Erzeugt die XML-Serialisierung der WCM-Filter-Objekte. Da es in VBA keine Vererbung gibt, ist die Implementierung etwas aufwendig.	`*Filter` (sämtliche WCM-Filter)
`User`	Serialisiert und deserialisiert die WCM-`User`-Objekte	`User`
`Value`	Serialisiert und deserialisiert die WCM-`Value`-Objekte	`*Value (StringValue, DateValue, LongValue ...)`
`Version`	Serialisiert und deserialisiert die WCM-`Version`-Objekte	`Version`
`Locale`	Serialisiert und deserialisiert die Java-`Locale`-Objekte	`Locale`

Verbindungsaufbau (Modul `VipWebServiceClient`)

Der Web-Service der Website “InternetSite” steht unter der URL http://wcmserver.company.example/wcm/WebService/Port/InternetSite/edit zur Verfügung. Ersetzen Sie dabei bitte die einzelnen Bestandteile der URL durch die Werte in Ihrer Konfiguration.

Der `SoapConnector` benötigt die folgenden Properties:

- `EndPointURL`
- `AuthUser` (Benutzerkennung des WCM-Benutzers, der auf die Daten zugreifen soll)
- `AuthPassword` (Passwort des WCM-Benutzers)

```
'
' VipWebServiceClient
'
' SOAP Encoding Namespace URI
Private Const SOAP_ENCODING_NS_URI = _
                         "http://schemas.xmlsoap.org/soap/encoding/"
Private Const VIP_NAMESPACE_URI = "http://gaussvip.com/"

' instance variables
Private serializer As SoapSerializer
Private Connector  As SoapConnector

'   ------------------------------------------------------------
'   initialize_connection
'   ------------------------------------------------------------
Private Sub initialize_connection()
 If Connector Is Nothing Then
   Set Connector = New HttpConnector
   ' set "EndPointURL"
   Connector.Property("EndPointURL") = _
"http://wcmserver.company.example/wcm/WebService/Port/InternetSite/
edit"
   ' set "SoapAction", "AuthUser" and "AuthPassword"
   Connector.Property("SoapAction") = VIP_NAMESPACE_URI
   Connector.Property("AuthUser") = "admin"
   Connector.Property("AuthPassword") = "a"
 Else
```

```
  Connector.Reset
End If
' initialise the SoapConnector Objekt and prepare the connection
Connector.Connect

' Begin the SOAP Message
Connector.BeginMessage

' create the necessary SoapSerializer
Set serializer = New SoapSerializer

' connect the Serializer with the SOAP Connection
serializer.Init Connector.InputStream

' Begin the SOAP Envelope
serializer.startEnvelope

' add the following required XML Namespace definitions:
serializer.SoapAttribute "xmlns:SOAP-ENC", , SOAP_ENCODING_NS_URI
serializer.SoapAttribute "xmlns:xsi", , _
                         "http://www.w3.org/2001/XMLSchema-instance"
serializer.SoapAttribute "xmlns:xsd", , _
                         "http://www.w3.org/2001/XMLSchema"
serializer.SoapAttribute "xmlns:vip", , VIP_NAMESPACE_URI
' Begin the request body
serializer.startBody
End Sub
```

Hinweise:

Das SOAP-Toolkit unterstützt leider keine Session-Verwaltung über Cookies. WCM WebServices führt deshalb für jeden Request eine separate Anmeldung durch. Für alle Funktionen, die Daten verändern (zum Beispiel “Vorlegen”) wird eine gültige WCM-Benutzerlizenz benötigt. WCM WebServices gibt diese Lizenz mit Ablauf der Session automatisch wieder frei – spätestens aber nach Ablauf des für den Administrationsserver eingestellten *Ablaufintervalls* nach der letzten feststellbaren Aktivität des Benutzers.

Informationen zur Konfiguration des Ablaufintervalls finden Sie im Livelink WCM Server-Administratorhandbuch. Zur Vermeidung von Lizenzproblemen sollten Sie die Methode `logout` von WCM WebServices zur expliziten Lizenzfreigabe verwenden.

Aufruf einer Methode ohne Rückgabewert

Nach der Initialisierung des `SoapSerializer` kann direkt die Methode mitsamt ihren Parametern in den Datenstrom eingefügt werden. Dies wird im folgenden Code-Beispiel am Beispiel der Methode `submit` (*Vorlegen*) gezeigt:

```
'   Submit an array of WCM objects (identified by their ObjectIDs) to QA
Public Function submit(ByRef oids() As ObjectID, _
                       ByVal pendingReleaseDate As Date, _
                       ByVal remark As String, _
                       ByVal emailI As emailInfo, _
                      ByVal dplWaitInfo As DeploymentWaitInfo) As Boolean
Dim success As Boolean
success = False
initialize_connection
' serialize request for submit
serializer.startElement "submit", VIP_NAMESPACE_URI

' Start element "oids"
'
serializer.startElement "oids"
    serializer.SoapAttribute "xsi:type", , "SOAP-ENC:Array"
    serializer.SoapAttribute "SOAP-ENC:arrayType", , _
             "vip:ObjectId[" & (UBound(oids) + 1 - LBound(oids)) & "]"
    Dim arrayOfOids As New Value
    Let arrayOfOids.ArrayValue = oids
    serializer.writeXML arrayOfOids.xml
serializer.endElement
' Start element "pendingReleaseDate"
'
serializer.startElement "pendingReleaseDate"
    Dim v As New Value
    Let v.dateValue = pendingReleaseDate

    ' pendingReleaseDate is only valid if it's in the future
    If pendingReleaseDate > Now Then
        serializer.SoapAttribute "xsi:type", , "xsd:dateTime"
        serializer.writeString v.simpleXml
    Else
        serializer.SoapAttribute "xsi:nil", , "true"
    End If
serializer.endElement
' Start element "remark"
'
serializer.startElement "remark"
```

```
    serializer.SoapAttribute "xsi:type", , "xsd:string"
    serializer.writeString remark
serializer.endElement
' Start element "emailInfo"
'
serializer.startElement "emailInfo"
    If Not emailI Is Nothing Then
        serializer.SoapAttribute "xsi:type", , "vip:EMailInfo"
        serializer.writeXML emailI.xml
    Else
        serializer.SoapAttribute "xsi:nil", , "true"
    End If
serializer.endElement
' Start element "dplWaitInfo"
'
serializer.startElement "dplWaitInfo"

    If Not dplWaitInfo Is Nothing Then
        serializer.SoapAttribute "xsi:type", , "vip:DeploymentWaitInfo"
        serializer.writeXML dplWaitInfo.xml
    Else
        serializer.SoapAttribute "xsi:nil", , "true"
    End If
serializer.endElement

serializer.endElement
' end of SOAP-Body
serializer.endBody
' end of SOAP-Envelope
serializer.endEnvelope
' end of SOAP message
Connector.EndMessage
' create a new SoapReader
Dim Reader As New SoapReader
' Load XML from OutputStream & display an error on failure
If Not Reader.Load(Connector.OutputStream) Then
    Dim reason As String
    reason = IIf(Reader.dom.parseError Is Nothing, _
              "parser: unknown reason", _
               Reader.dom.parseError.reason)
    MsgBox "submit failed:- " & reason, vbExclamation, "Error"
    Exit Function
End If
' if there is an application fault use a message box to report it
If Not Reader.Fault Is Nothing Then

    MsgBox "submit failed:- " & _
            Reader.faultstring.nodeTypedValue,_
            vbExclamation, "Error"
```

```
ElseIf Not Reader.Body Is Nothing Then
    submit = True
End If

End Function
```

Diese Funktion zeigt sehr anschaulich, wie die Parameter nach XML serialisiert werden müssen:

- das Array mit den OIDs wird über die `Value`-Klasse serialisiert
- `EMailInfo` und `DeploymentWaitInfo` sind eigene Klassen, die jeweils eine `xml()`-Funktion beinhalten
- das Bemerkungsfeld (vom Typ `String`) wird hier direkt ausgegeben
- für das Datumsfeld wird die `Value`-Klasse verwendet

Alle fehlerhaften bzw. fehlenden Parameter werden durch `xsi:nil="True"` repräsentiert. Sie können im Prinzip auch weggelassen werden, da Web-Services die Parameter über ihre Namen identifizieren und fehlende Parameter durch die Typ-spezifischen Standardwerte (`False` für `Boolean`, 0 für `int` und `long` sowie `null` für komplexe Typen) ersetzen. Das Einlesen der Serverantwort muss sorgfältig erfolgen. Es sind folgende Fehlerszenarien denkbar:

- der Server antwortet gar nicht
- die Antwort des Servers enthält kein XML
- die Antwort des Servers enthält keinen SOAP-Envelope
- die Antwort des Servers enthält keinen SOAP-Body
- die Antwort des Servers enthält einen Anwendungsfehler

Im Beispielprogramm werden genau diese Aspekte getestet. Der Rückgabewert der Funktion `submit(...)` zeigt an, ob das Vorlegen zur Qualitätssicherung erfolgreich durchgeführt wurde.

Der Schwerpunkt des nächsten Beispiels ist die Deserialisierung der vom Web-Service gelieferten XML-Daten.

Aufruf einer Methode mit Rückgabewert

Als Beispiel wird hier die Auswertung eines Filterausdrucks verwendet. WCM WebServices liefert als Ergebnis ein Array von `ObjectData`. Dieses Array besteht aus `n` Einträgen, die ihren Inhalt über `href="#id..."` referenzieren. In dem Beispiel werden XPath-Ausdrücke verwendet, um die entsprechenden Elemente im DOM-Tree zu lokalisieren.

`ObjectData` selber ist ein komplexer Datentyp, der unter anderem Elemente des Typs `ObjectState`, `ObjectType`, `User`, `ArrayOfString` und `ArrayOfObjectId` referenziert. Diese sind auch wieder über `href="#id..."` eingebunden.

Zur Deserialisierung eines solchen komplexen Objekts mittels einer einfachen Programmiersprache gibt es mehrere Wege. Hier wird folgender Ansatz genutzt:

1. Die SOAP-Antwort wird in einen separaten Parser geladen.

```
resultDom.loadXML(Reader.Body.xml)
```

2. Dort wird das Attribut `href` im Element `filterResult` gesucht.

```
result = resultDom.selectSingleNode("//filterResult/@href")
```

3. Aus dem Attribut wird der Wert der entsprechenden ID extrahiert.

```
resultId = Mid(result.text, 2)
```

4. Dadurch wird das Array im DOM lokalisiert.

```
List = resultDom.selectNodes("//*[@id='" & resultId & "']/*")
```

5. Für jeden Eintrag in der Liste wird nun ein `ObjectData`-Objekt angelegt.

```
itemId = _
    Mid(List.Item(i).Attributes.getNamedItem("href"). nodeValue, 2)
objectDataItem = resultDom.selectNodes("//*[@id='" &_
                                    itemId & "']/*")
results(i) = tmpObjectData.getObjectFromNodeList(_
                         objectDataItem, resultDom)
```

6. Der Methode `getObjectFromNodeList` werden die einzelnen `ObjectData`-Attribute als eine `IXMLDOMSelection` übergeben.

   Diese Elemente enthalten jeweils die Objekttyp-Information (zum Beispiel `<title xsi:type="xsd:string"> PortalManager-ger_v01</title>`). Abhängig vom Typ wird nun das entsprechende Attribut in `ObjectData` gesetzt. Komplexe Typen wie zum Beispiel `ObjectState` verfügen jeweils über eine eigene Methode `getObjectFromNodeList(...)`.

Hier folgt der Code der Methode `filter`:

```
'
' filter: returns an array of ObjectData containing the objects
' matching the specified filter and READ permissions for the
' authenticated user. The number of records returned is passed
' via the reference variable numRows.
'
Public Function filter(ByVal myFilter As GenericFilter, _
                       ByVal startOid As ObjectID, _
                       ByRef sortList() As Sort, _
                       ByVal startResult As Long, _
                       ByVal numberOfResults As Long, _
                       ByRef attributeKeys() As String, _
                       ByRef numRows As Integer) As Variant
```

```
initialize_connection
    numRows = 0
    ' serialize request for filter
    serializer.startElement "filter", VIP_NAMESPACE_URI

    ' Start element "filter"
    ' (xsi:type="vip:[EqualFilter|LessFilter ...])
    '
    serializer.startElement "filter"
     serializer.SoapAttribute "xsi:type", , "vip:" & myFilter.FilterType

     If Not myFilter Is Nothing Then
        serializer.writeXML myFilter.xml(serializer)
     End If
    serializer.endElement
    ' Start element "startOID"
    '
    serializer.startElement "startOID"
     serializer.SoapAttribute "xsi:type", , "vip:ObjectId"
     If Not startOid Is Nothing Then
            serializer.writeXML startOid.xml
     End If
    serializer.endElement
    ' Start element "sortList"
    '
    serializer.startElement "sortList"
     serializer.SoapAttribute "xsi:type", , "SOAP-ENC:Array"
     serializer.SoapAttribute "SOAP-ENC:arrayType", ,_
             "vip:Sort[" & (UBound(sortList) - LBound(sortList)) & "]"

     Dim i As Integer

     For i = LBound(sortList) To UBound(sortList) - 1
         serializer.startElement "Item"
             serializer.SoapAttribute "xsi:type", , "vip:Sort"
             serializer.writeXML sortList(i).xml
         serializer.endElement
     Next i
    serializer.endElement
    ' Start element "startResult"
    '
    serializer.startElement "startResult"
     serializer.SoapAttribute "xsi:type", , "xsd:int"
     serializer.writeString startResult
    serializer.endElement
    ' Start element "numberOfResults"
    '
    serializer.startElement "numberOfResults"
     serializer.SoapAttribute "xsi:type", , "xsd:int"
```

```
 serializer.writeString numberOfResults
serializer.endElement
'
' start attributeKeys
serializer.startElement "attributeKeys"
 serializer.SoapAttribute "xsi:type", , "SOAP-ENC:Array"
 serializer.SoapAttribute "SOAP-ENC:arrayType", , _
         "xsd:string[" & (UBound(attributeKeys) -_
                          LBound(attributeKeys)) & "]"

 For i = LBound(attributeKeys) To UBound(attributeKeys) - 1
     serializer.startElement "Item"
         serializer.SoapAttribute "xsi:type", , "xsd:string"
         serializer.writeXML attributeKeys(i)
     serializer.endElement
 Next i
serializer.endElement

serializer.endElement
' close SOAP-Body
serializer.endBody
' close SOAP-Envelope
serializer.endEnvelope

' signal the end of the SOAP message being sent to the server
Connector.EndMessage

' create a new SoapReader
Dim Reader As New SoapReader

' load the XML from the Stream
If Not Reader.Load(Connector.OutputStream) Then
    Dim reason As String
    reason = IIf(Reader.dom.parseError Is Nothing, _
                 "parser: unknown reason", _
                 Reader.dom.parseError.reason)

    MsgBox "filter failed:- " & reason, vbExclamation, "Error"
    Exit Function
End If

' if there is a fault use a message box to report it
If Not Reader.Fault Is Nothing Then
    MsgBox "filter failed:- " & _
            Reader.faultstring.nodeTypedValue,_
            vbExclamation, "Error"
Else
    ' get the return Document
    Dim resultDom As DOMDocument30
```

```
Set resultDom = New DOMDocument30
resultDom.setProperty "SelectionLanguage", "XPath"
resultDom.async = False
resultDom.validateOnParse = False

Dim success As Boolean
' set up the temporary results array
Dim results() As objectData

If Reader.Body Is Nothing Then
  success = False
Else
  Let success = resultDom.loadXML(Reader.Body.xml)

  ' get the nodelist of array items from the response
  ' traverse to multiref with corresponding id
  Dim result As IXMLDOMNode
  Dim resultId As String

  ' look for corresponding attribute href=<xyy>
  Set result = resultDom.selectSingleNode(_
                                    "//filterResult/@href")

  If Not result Is Nothing Then
    resultId = result.text
    resultId = Mid(resultId, 2)

    Dim List, objectDataItemList As IXMLDOMNodeList

   Set List = resultDom.selectNodes("//*[@id='"&resultId&"']/*")

    If List.Length = 0 Then
        Exit Function
    End If

    ReDim results(List.Length - 1)
    numRows = List.Length

    ' for each array element
    For i = 0 To (List.Length - 1)
      ' Create Temporary "DocData"
      Dim tmpDocData As New objectData

      Dim itemId As String
      itemId =_
  Mid(List.Item(i).Attributes.getNamedItem("href").nodeValue, 2)
```

```
            Set objectDataItemList =_
                  resultDom.selectNodes("//*[@id='" & itemId & "']/*")
            ' Populate the array element
            Set results(i) =_
                 tmpDocData.getObjectFromNodeList(objectDataItemList,_
                                                        resultDom)
            Next

          End If
        End If
        ' set the return value
        filter = results
    End If
End Function
```

Die Filter-Methode kann zum Beispiel folgendermaßen angewendet werden:

```
Sub vipFilterPDF()
    Dim filteresult As Variant

    If VipWebService Is Nothing Then
        Set VipWebService = New VipWebServiceClient
    End If

    Dim pdfFilter As New GenericFilter

    Let pdfFilter.FilterType = "EqualFilter"
    Set pdfFilter.Key = New Key
    Let pdfFilter.Key.stringValue = "objectType"
    Set pdfFilter.Value = New Value
    Set pdfFilter.Value.objectType = New objectType
    Let pdfFilter.Value.objectType.name = "PDF"

    Dim startId As New objectID
    Let startId.id = "1"

    Dim dateSort(1) As New Sort
    Let dateSort(0).stringValue = "createdDate"

    Dim attributeKeys(4) As String
    Let attributeKeys(0) = "createdDate"
    Let attributeKeys(1) = "objectId"
    Let attributeKeys(2) = "objectState"
    Let attributeKeys(3) = "title"
```

```
    Dim numRows As Integer
    filterResult = VipWebService.filter(pdfFilter, startId, dateSort, _
                                        0, 5, attributeKeys, numRows)

End Sub
```

Mit diesem Beispiel werden die Code-Beispiele für Visual Basic beendet. Den vollständigen Quellcode für die in Tabelle 74 “Klassenmodule” auf Seite 163 aufgelisteten Module finden Sie im Verzeichnis **{WCM-Installationsverzeichnis}\examples\webservices\vba**. Diese Klassen können Sie über *Datei importieren* in ein Visual Basic Projekt übernehmen.

6.2 C# und ASP.NET

Die Entwicklung Browser-basierter Benutzeroberflächen mit ASP.NET oder Standard-Windows-Oberflächen mit Windows.Forms wird durch Visual Studio .NET sehr vereinfacht. In diesem Abschnitt erfahren Sie, wie Sie WCM WebServices in Visual Studio nutzen können.

Vorarbeiten

Richten Sie ein neues C#-Projekt ein, z.B. mit dem Namen “VipWebServiceClient”:

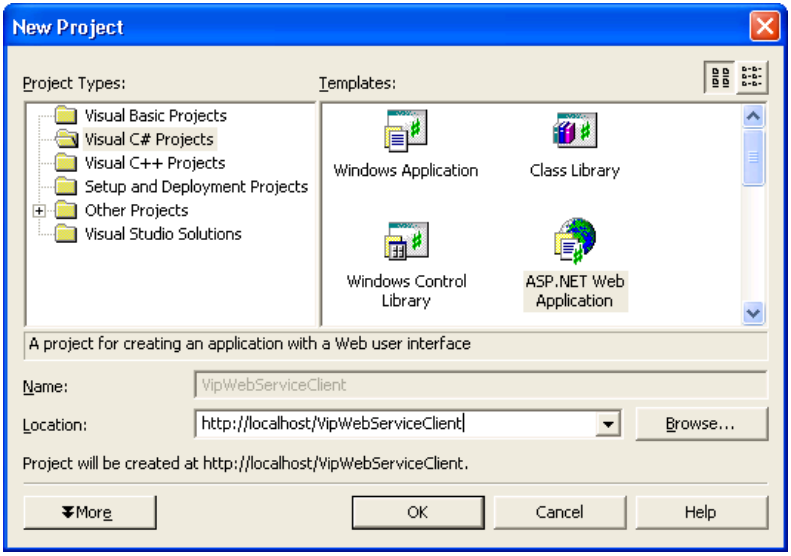


Abb. 9 – C#-Projekt einrichten

Visual Studio erzeugt automatisch die benötigten Module für den Zugriff auf WCM WebServices, sobald Sie die URL der WSDL-Beschreibung als Webverweis (Im Menü *Projects* → *Add Web Reference*) hinzufügen:

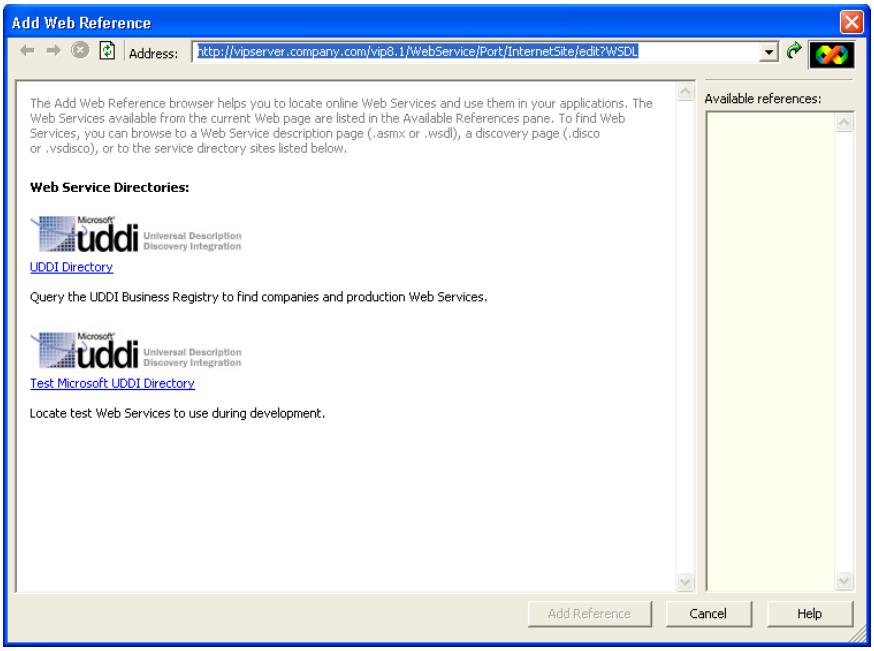

Add Web Reference
Address:
http://vipserver.company.com/vip8.1/WebService/Port/InternetSite/edit?WSDL
The Add Web Reference browser helps you to locate online Web Services and use them in your applications. The Web Services available from the current Web page are listed in the Available References pane. To find Web Services, you can browse to a Web Service description page (.asmx or .wsdl), a discovery page (.disco or .vsdisco), or to the service directory sites listed below.
Web Service Directories:
Microsoft uddi Universal Description Discovery Integration
UDDI Directory
Query the UDDI Business Registry to find companies and production Web Services.
Microsoft uddi Universal Description Discovery Integration
Test Microsoft UDDI Directory
Locate test Web Services to use during development.
Available references:
Add Reference
Cancel
Help

Abb. 10 – WSDL als Webverweis hinzufügen

Visual Studio lädt das WSDL in den linken Bereich dieses Fensters und verlangt dann noch einmal Benutzernamen und Passwort:

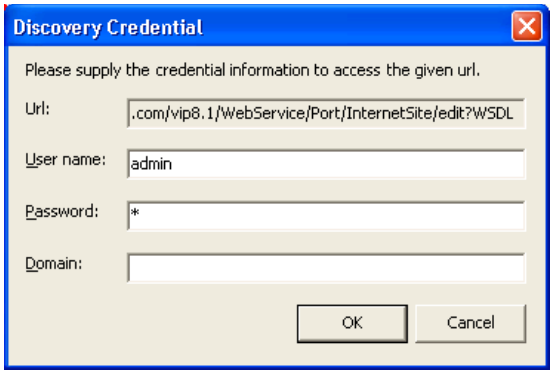


Abb. 11 – Benutzerkennung und Passwort eingeben

Nach erfolgreicher Autorisierung können Sie den Dialog *Add Web Reference* abschließen, indem Sie auf *Add Reference* klicken.

Visual Studio erzeugt aus der WSDL-Beschreibung von WCM WebServices ein Modul im Namensraum (*Namespace*) `VipWebServiceClient.com.company.wcmserver`. Dort finden Sie für jeden in WCM WebServices benötigten Datentyp eine Klasse. Der *Object Browser* gibt einen guten Überblick über die automatisch erzeugten Klassen:

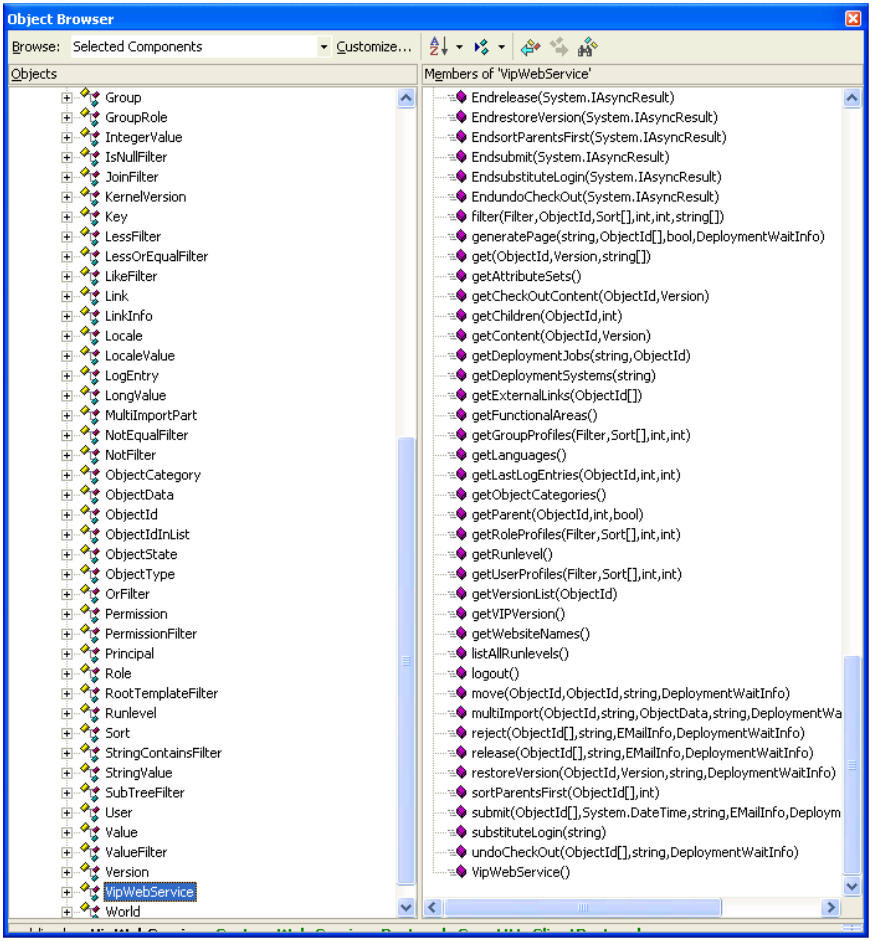


Abb. 12 – Automatisch erzeugte Klassen im Namensraum `VipWebServiceClient`

Sie können nun mit der Nutzung von WCM WebServices beginnen. Als Beispiel wird im Folgenden eine ASP.NET-Seite (**default.aspx**) verwendet, die in einer Tabelle alle PDF-Dateien der Website “Internet-Site” auflistet. Jede Zeile der Tabelle enthält ein Kästchen zur Auswahl der Zeile. Alle ausgewählten Objekte lassen sich über eine Schaltfläche vorlegen bzw. freigeben.

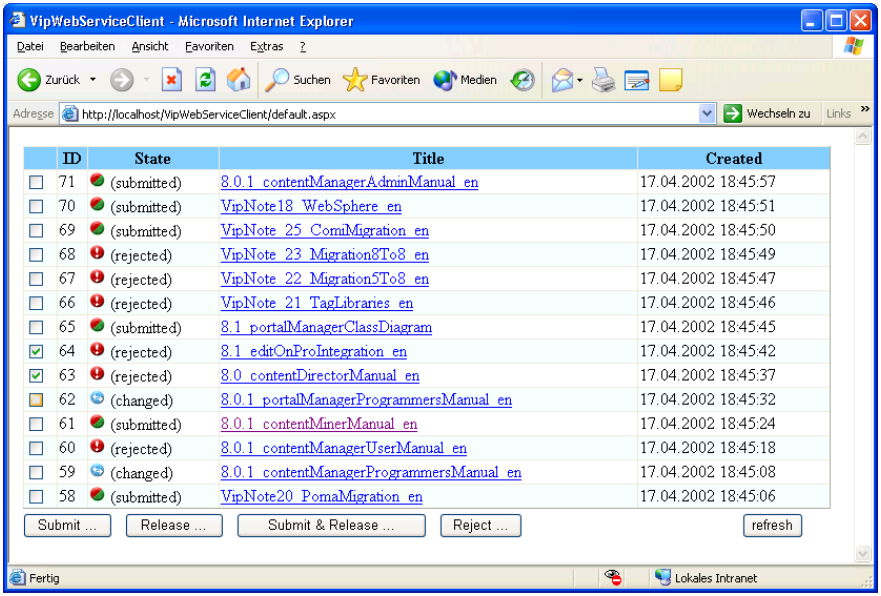

VipWebServiceClient - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Zurück Suchen Favoriten Medien
Adresse http://localhost/VipWebServiceClient/default.aspx
Wechseln zu Links
ID State Title Created
71 (submitted) 8.0.1_contentManagerAdminManual_en 17.04.2002 18:45:57
70 (submitted) VipNote18_WebSphere_en 17.04.2002 18:45:51
69 (submitted) VipNote_25_ComiMigration_en 17.04.2002 18:45:50
68 (rejected) VipNote_23_Migration8To8_en 17.04.2002 18:45:49
67 (rejected) VipNote_22_Migration5To8_en 17.04.2002 18:45:47
66 (rejected) VipNote_21_TagLibraries_en 17.04.2002 18:45:46
65 (submitted) 8.1_portalManagerClassDiagram 17.04.2002 18:45:45
64 (rejected) 8.1_editOnProIntegration_en 17.04.2002 18:45:42
63 (rejected) 8.0_contentDirectorManual_en 17.04.2002 18:45:37
62 (changed) 8.0.1_portalManagerProgrammersManual_en 17.04.2002 18:45:32
61 (submitted) 8.0.1_contentMinerManual_en 17.04.2002 18:45:24
60 (rejected) 8.0.1_contentManagerUserManual_en 17.04.2002 18:45:18
59 (changed) 8.0.1_contentManagerProgrammersManual_en 17.04.2002 18:45:08
58 (submitted) VipNote20_PomaMigration_en 17.04.2002 18:45:06
Submit ... Release ... Submit & Release ... Reject ... refresh
Fertig
Lokales Intranet

Abb. 13 – Die Beispielanwendung (Liste aller PDF-Dateien)

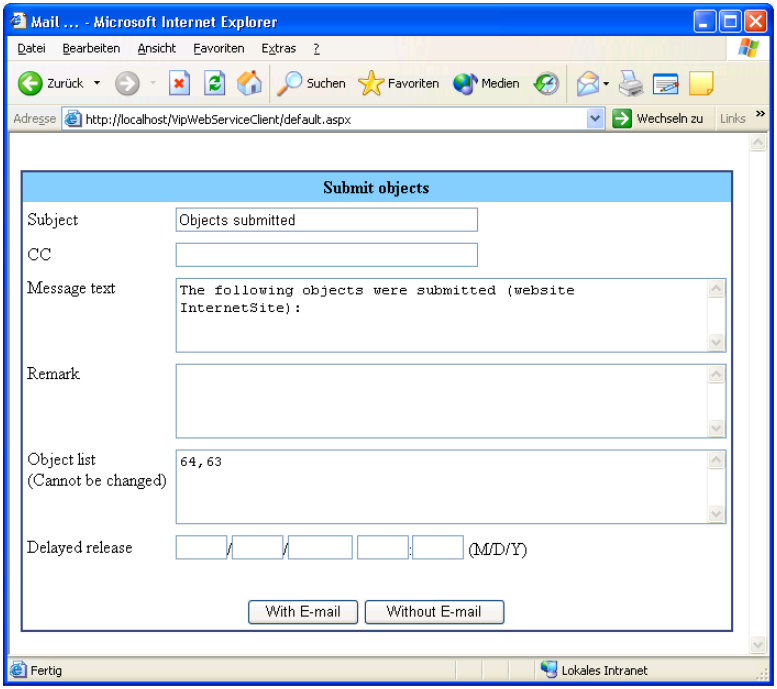


Abb. 14 – Maildialog der Beispielanwendung

Dieses Beispiel besteht aus den folgenden Dateien:

Tabelle 75 – ASP.NET-Beispiel

Name	Beschreibung	Verwendete Elemente

default.aspx	ASP-Seite: Benutzeroberfläche
	Enthält ein `DataGrid` zur Anzeige der Objektdaten
		`<asp:datagrid>`
		`<asp:label>`
		`<asp:TemplateColumn>`
		`<asp:CheckBox>`
		`<asp:HyperLink>`
		`<asp:button>`

Name	Beschreibung	Verwendete Elemente

default.aspx.cs	Anwendungscode hinter der ASP-Seite. Hier werden die Daten aus Livelink WCM Server geladen. Außerdem sind die Callbacks für die fünf Schaltflächen implementiert.	`CookieContainer`
		`CredentialCache`
		`NetworkCredential`
		`Server`
		`Session`
		`EqualFilter`
		`VipWebService`

mail.aspx	Einfaches Mail-Erfassungsformular (siehe Abbildung 14 “Maildialog der Beispielanwendung” auf Seite 182). Hier können zusätzliche Mailempfänger eingetragen, das Freigabedatum vorgegeben und Anmerkungen für das Objektprotokoll eingetragen werden.	`<asp:label>`
		`<asp:literal>`
		`<tr runat="server">`

mail.aspx.cs	Der Code hinter dem Mailformular. Insbesondere wird die Zeile mit dem Freigabedatum nur sichtbar, wenn der Dialog zum Vorlegen zur QS genutzt wird. Im Fehlerfall wird unten eine Zeile mit der Fehlerbeschreibung eingeblendet.	`VipWebService` `SoapException`

Verbindungsaufbau

Die korrekte URL für den Verbindungsaufbau wird von WCM WebServices in das WSDL eingetragen und von Visual Studio in den automatisch erzeugten Code eingefügt. Ihre Aufgabe ist es, den Code für die Authentifizierung und das Session-Tracking einzufügen:

```
/// <summary>
/// Initializes the service and saves the service - reference within the
/// sesion (as 'service')
/// </summary>
private void intializeService()
{
      service = (VipWebService)Session["service"];
      if (service == null)
      {
         service = new VipWebService ();
         // allow reception of cookies
         service.CookieContainer = new CookieContainer();
         // create the necessary credentials
         // (using basic authentication)
         NetworkCredential credentials =
                  new NetworkCredential (userName, userPassword);

         CredentialCache credCache = new CredentialCache ();
         credCache.Add(new Uri(service.Url), "Basic", credentials);
         service.Credentials = credCache;

         Session["service"] = service;
         Session.Timeout = 60;
      }
}
```

Hinweis: Das .NET-Framework hat eine voreingestellte Zeitbegrenzung für synchrone Web-Service-Aufrufe. Der Standardwert beträgt 3 Minuten. Mit Ausnahme aufwendiger Importe von verknüpften Objekten sind WCM WebServices-Aufrufe normalerweise innerhalb kürzerer Zeit beendet. Sie können die Zeitbegrenzung mithilfe des Attributs `service.Timeout` (in Millisekunden) auf einen beliebigen anderen Wert einstellen. So stellt `service.Timeout = 60000 * 10;` die Zeitbegrenzung auf 10 Minuten ein.

Aufbau der Seite

Beim Laden der ASP-Seite wird die Methode Page_Load aufgerufen. Hier wird der Filter über den Web-Service ausgeführt:

```
private void Page_Load(object sender, System.EventArgs e)
{
      // Put user code to initialize the page here
      // initialize the service ...
      intializeService();

      if (! IsPostBack)
      {
         // load the data into the DataGrid
         tableOfObjects.DataSource = filterPDF();
         tableOfObjects.DataBind();
      }
}
```

Die eigentliche Filterung ist dabei in die folgende Methode ausgelagert:

```
/// <summary>
/// Retrieves all PDF files via a WCM Web Service call
/// </summary>
/// <returns></returns>
private ObjectData[] filterPDF()
{
      EqualFilter filter = new EqualFilter ();
      filter.key   = new Key();
      filter.value = new ObjectType();

      filter.key.stringValue = "objectType";
      ((ObjectType)filter.value).name = "PDF";

      ObjectId startId = new ObjectId ();
      startId.id = "1";

      Sort[] sortList = new Sort[]{ new Sort() };
      sortList[0].stringValue = "createdDate";
      sortList[0].descending  = true;

      String[] attributeKeys = new String []{   "objectId",
                                        "objectState",
                                        "title",
```

```
                                        "createdDate",
                                        "createdBy",
                                        "modifiedDate",
                                        "modifiedBy",
                                        "URL"};

      return service.filter (filter, startId, sortList,
                             0, 100, attributeKeys);
}
```

Dieser Code zeigt noch einmal, wie einfach es ist, die Methoden von WCM WebServices zu nutzen. Durch die “CodeCompletion”-Technik in Visual Studio haben Sie sofort einen Überblick über die verfügbaren Methoden, Datentypen und Attribute.

Aufbau der Tabelle

Zur Darstellung der Filterergebnisse wird ein `DataGrid`-Control verwendet. Es bietet eine ausreichende Flexibilität in der Darstellung der Daten.

Für die Präsentation der Daten sorgen so genannte `TemplateColumns`. Im `ItemTemplate` hat man vollen Zugriff auf den Inhalt der aktuellen Tabellenzeile (hier ein `ObjectData`-Objekt). Damit die Darstellung der Ergebnisse funktioniert, muss die Eigenschaft `AutoGenerateColumns` des `DataGrid` auf `False` gesetzt werden.

So sehen die `ItemTemplates` aus:

```
<asp:TemplateColumn HeaderText="ID">
      <ItemTemplate>
         <asp:Label id="C_OID" runat="server"
        Text="<%#((ObjectData)Container.DataItem).objectId.id%>">
         </asp:Label>
      </ItemTemplate>
</asp:TemplateColumn>
```

In einer Zelle der Tabelle können aber auch mehrere Elemente verwendet werden: hier z.B. das Symbol für den Objektstatus gefolgt von der Beschreibung:

```
<asp:TemplateColumn HeaderText="State">
  <ItemTemplate>
    <IMG src="<%#((ObjectData)Container.DataItem).objectState.imageURL
              %>">
        (<%#((ObjectData)Container.DataItem).objectState.description %>)
  </ItemTemplate>
</asp:TemplateColumn>
```

Die erste Spalte der Tabelle beinhaltet eine `CheckBox`. Sie ist hier als serverseitiges Steuerelement realisiert. Jeder Klick in eine `CheckBox` wird über die `PostBack`-Funktion direkt auf den Server übertragen:

```
<asp:TemplateColumn>
      <ItemTemplate>
         <asp:CheckBox id="cb1" runat="server" AutoPostBack="True">
          </asp:CheckBox>
      </ItemTemplate>
</asp:TemplateColumn>
```

Auf dem Server wird aus allen durch die `CheckBox` ausgewählten Zeilen die `ObjectId` extrahiert und zur weiteren Verarbeitung an den Maildialog übergeben (**mail.aspx**).

Aufbau des Maildialogs

Der Maildialog unterscheidet sich für die verschiedenen Staging-Aktionen (Vorlegen, Freigeben, Direkte Freigabe, Ablehnen) nur durch Folgendes:

- die Texte sind unterschiedlich
- die Eingabe des gewünschten Freigabedatums ist beim Vorlegen bzw. bei der direkten Freigabe möglich

Die entsprechende Zeile der Tabelle ist deshalb als serverseitiges Steuerelement (`id="trDelayed"`) realisiert und wird in der Methode `Page_Load` entsprechend ein- bzw. ausgeblendet:

```
String operation = Request.Params["operation"];
trDelayed.Visible = (   operation.Equals("submit")
                     || operation.Equals("directRelease"));
```

Ausführen der Aktion

Sobald im Maildialog eine der beiden Schaltflächen ausgewählt wird, wird die gewünschte Aktion ausgeführt. Zu diesem Zweck werden die Eingabedaten aus dem Dialog in ein `EMailInfo`-Objekt übernommen und der gewünschten Methode als Parameter übergeben:

```
private void execute(Boolean withMail)
{
      VipWebService service = (VipWebService) this.Session["service"];
      String operation = Request.Params["operation"];

      if (service != null)
      {
         String[] oids =
                     textAreaObjectList.Value.Split(",".ToCharArray());

         ObjectId[] oidsToSubmit = new ObjectId[oids.Length];
         for (int i = 0; i < oids.Length; i++)
         {
            oidsToSubmit[i] = new ObjectId ();
            oidsToSubmit[i].id = oids[i];
         }
         DateTime pendingReleaseDate;

         pendingReleaseDate = new DateTime(1,1,1);

         EMailInfo eMailInfo = null;
         if (withMail)
         {
            eMailInfo         = new EMailInfo ();
            eMailInfo.subject = textAreaMessageText.Value;
            eMailInfo.cc      = textAreaCC.Value;
         }
```

```
        DeploymentWaitInfo dplWaitInfo = null;
        //
        try
        {
           switch(operation)
           {
              case "submit":
                 service.submit (oidsToSubmit, pendingReleaseDate,
                                 textAreaRemark.Value, eMailInfo,
                                 dplWaitInfo);
                 break;

              case "directRelease":
                 service.directRelease (oidsToSubmit,
                                        pendingReleaseDate,
                                        textAreaRemark.Value,
                                        eMailInfo,
                                        dplWaitInfo);
                 break;

              case "release":
                 service.release (oidsToSubmit,
                                  textAreaRemark.Value, eMailInfo,
                                  dplWaitInfo);
                 break;

              case "reject":
                 service.reject (oidsToSubmit,
                                 textAreaRemark.Value, eMailInfo,
                                 dplWaitInfo);
                 break;
           }
           // ... and return to the calling page
           Server.Transfer (nextURL);
        }
        catch (Exception exc)
        {
           Exception e1 = exc;
           rowError.Visible = true;
           literalError.Text = exc.Message;
        }
     }
}
```

Die gewünschte Aktion wird in einem Try-Catch-Block ausgeführt. Hier werden alle Ausnahmebedingungen abgefangen.

Fehlerbehandlung

Es liegt in der Natur verteilter Anwendungen, dass es mehr mögliche Fehlerquellen gibt. Umso sorgfältiger müssen Sie als Anwendungsentwickler die Fehlerbedingungen auswerten und in einer Form präsentieren, die für den Endbenutzer verständlich ist.

Zu den möglichen Fehlerquellen gehören:

- Verbindungsprobleme zu WCM WebServices (WCM-Server nicht erreichbar)
- Authentifizierungsprobleme (falscher Benutzer oder falsches Passwort)
- Ablauf der Zeitbegrenzung für den Aufruf (Timeout zu klein)
- SOAP-Fehler (Client oder Server schicken fehlerhafte bzw. nicht korrekt interpretierbare SOAP-Envelopes)
- Fehlermeldungen des WCM-API (Operation nicht erlaubt, Systemfehler wie zum Beispiel Fehler beim Zugriff auf die Datenbank)

Alle oben genannten Fehler werden durch eine `Message` in der geworfenen `Exception` beschrieben. Diese wird in dem hier dargestellten Maildialog angezeigt.

Die Fehlermeldungen von WCM WebServices werden automatisch in der Sprache des Benutzers ausgegeben. Neben dem kurzen, für den Endbenutzer gedachten Fehlertext enthalten die Fehlermeldungen von Livelink WCM Server weitere Detailinformationen, die speziell für den Systemadministrator und den Technical Support der Gauss Interprise AG wichtig sind. Diese können – ähnlich wie im Content-Client – als Details angezeigt werden.

Der Fehlercode steht über das Attribut `Code` (vom Typ `System.Xml.XmlQualifiedName`) der `SoapException` zur Verfügung. Eine Liste der Server-Fehlercodes finden Sie im Abschnitt “Exceptions des Content-Servers” auf Seite 49.

Sämtliche Detailinformationen sind über das Attribut `Detail` (vom Typ `System.Xml.XmlNode`) der `SoapException` verfügbar.

Der folgende Codeauszug enthält die SOAP-Message mit dem Fehlercode:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
            xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
            xmlns:tns="http://gaussvip.com/"
            xmlns:types="http://gaussvip.com/encodedTypes"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <soap:Fault>

    <faultcode xmlns:ns1="http://gaussvip.com/">
                  ns1:Server.ActionNotPermittedException
    </faultcode>
    <faultstring>
         0582: No direct release is permitted for object &apos;68&apos;.
    </faultstring>
    <faultactor>Vip-Content-Manager-API
    </faultactor>
    <detail>

     <tns:message sequenceNo="0">
         0303: The master server was unable to perform the request.
     </tns:message>
     <tns:message sequenceNo="1">
         0738: Direct release of object &apos;68&apos; failed.
     </tns:message>
     <tns:message sequenceNo="2">
         0582: No direct release is permitted for object &apos;68&apos;.
     </tns:message>
     <tns:wrappedExceptionName>de.gauss.vip.api.VipApiException
     </tns:wrappedExceptionName>

     <tns:wrappedExceptionMessage>
         0303: The master server was unable to perform the request.
     </tns:wrappedExceptionMessage>
    </detail>
   </soap:Fault>
  </soap:Body>
 </soap:Envelope>
```

Eine Detailanzeige könnte also die <tns:message>-Elemente anzeigen. Dabei gilt die Regel, dass die Meldungen mit aufsteigender Seriennummer (Attribut sequenceNo) immer spezifischer werden.

So könnte der Quellcode aussehen:

```
catch (SoapException soapExc)
{
      rowError.Visible = true;
      literalError.Text = soapExc.Message
                     + "<hr/>Code: " + soapExc.Code.Name
                     + "<hr/>Details<br/>";

      XmlNodeList details = soapExc.Detail.ChildNodes;

      for (int i = 0; i < details.Count; i++)
      {
         XmlNode detail = details[i];
         if (detail.LocalName.Equals ("message"))
         {
            literalError.Text += "<br/>" + detail.FirstChild.Value;
         }
      }
      literalError.Text += "<hr/>";
}
catch (Exception exc)
{
      rowError.Visible = true;
      literalError.Text = exc.Message;
}
```

Damit sind die Code-Beispiele für C# und ASP.NET abgeschlossen. Den vollständigen Quellcode für die in Tabelle 75 “ASP.NET-Beispiel” auf Seite 182 aufgelisteten Module finden Sie im Verzeichnis **{WCM-Installationsverzeichnis}\examples\webservices\asp.net**. Diese Dateien können Sie über *Add existing item* in ein Visual Studio-Projekt übernehmen.

Caveats

Bei der Nutzung von WCM WebServices im .NET-Umfeld ist Folgendes zu beachten: Es gibt einen Namenskonflikt zwischen dem WCM-Datentyp `Version` und dem eingebauten .NET-Datentyp `Version` im Namensraum `System`. Aus diesem Grunde muss der WCM-Datentyp `Version` immer vollständig mit Namensraum qualifiziert werden.

Beispiel:

```
VipWebServiceClient.com.company.wcmserver.Version v =
             new VipWebServiceClient.com.company.wcmserver.Version();
  ...
```

Glossar

ACL – Access Control List, siehe *Zugriffssteuerungsliste*

Aktion – Schritt, der zur Verwaltung von Website-Objekten notwendig ist und meistens zur Statusänderung des betreffenden Objekts führt

Asynchrone Aktion – Eine asynchrone Aktion kehrt praktisch sofort nach dem Aufruf zurück, wartet also nicht, bis die damit verbundenen Aufgaben abgeschlossen sind. Siehe auch *synchrone Aktion*.

Ausleihen – Staging-Aktion von Livelink WCM Server: Der Inhalt eines WCM-Objekts kann erst bearbeitet werden, wenn das Objekt ausgeliehen wurde. Ein ausgeliehenes Objekt ist für die Zugriffe anderer Benutzer gesperrt. Die Änderungen, die am Objekt vorgenommen werden, sind erst nach dem Zurückgeben in der Edit-Sicht verfügbar.

Datenhaltungssicht – Die Datenhaltungssicht eines Servers bezeichnet die aktuell verfügbaren Ausprägungen der WCM-Objekte (Edit-Sicht, QS-Sicht, Produktionssicht). Über welche Datenhaltungssicht ein Server verfügt, wird in den Routing-Einstellungen der Website festgelegt.

Deployment – Deployment bezeichnet die Verteilung von Daten. Das Deployment von Livelink WCM Server übernimmt zwei Hauptaufgaben: erstens die Generierung von Seiten aus den in der Datenbank gespeicherten WCM-Objekten und die Verteilung der generierten Dateien in die dafür vorgesehenen Verzeichnisse; zweitens die Benachrichtigung der WCM-Server bei Änderungen im WCM-System.

Deploymentsystem – Die Deploymentsysteme erzeugen aus den WCM-Objekten Seiten und verteilen die generierten Dateien in die dafür vorgesehenen Verzeichnisse. Von dort aus werden die Dateien über den Einsatz eines HTTP-Servers für die Benutzer sichtbar. Deploymentsysteme können unterschiedliche Typen und Kategorien haben.

Deploymentsystem-Kategorie – Je nach Art der Verarbeitung von Deploymentaufträgen werden Deploymentsysteme verschiedenen Kategorien zugeordnet: *Standard-Deploymentsysteme* erzeugen bei jeder Änderung an einem WCM-Objekt automatisch eine neue Seite. Die generierten Seiten werden im Dateisystem abgelegt. *Dynamische Deploymentsysteme* generieren die Seiten auf Grundlage benutzerdefinierter Einstellungen und nur dann, wenn die Seite über den HTTP-Server angefordert wird. Die generierten Dateien werden in einer flachen Dateistruktur abgelegt. Mit *Suchmaschinen-Deploymentsystemen* können Sie die Daten Ihrer Website für den Einsatz einer Suchmaschine aufbereiten. *WebDAV-Deploymentsysteme* bilden die Voraussetzung für den Einsatz von WebDAV-Clients. *InSite Editing-Deploymentsysteme* schaffen die Voraussetzung dafür, dass Inhalte direkt innerhalb der Website – ohne Content-Client – bearbeitet und hinzugefügt werden können.

Deploymentsystem-Typen – Bei Deploymentsystemen wird entsprechend dem Staging-Konzept von Livelink WCM Server zwischen den Typen “Edit”, “QS” und “Produktion” unterschieden. Je nach Typ werden unterschiedliche Sichten auf die Daten der Website erzeugt.

Edit-Sicht – In der Edit-Sicht von Livelink WCM Server werden die Objekte einer Website angelegt und redaktionell bearbeitet. Hier ist der jeweils aktuelle Bearbeitungsstand zu sehen.

Freigeben – Staging-Aktion von Livelink WCM Server: Die Qualitätssicherung prüft inhaltlich und formal, ob ein vorgelegtes Objekt den Qualitätsstandards des Unternehmens entspricht. Ist dies der Fall, wird das Objekt freigegeben. Durch die Freigabe wird die qualitätsgesicherte Version des Objekts in die Produktionssicht übertragen und damit dem Endbenutzer in der publizierten Website verfügbar gemacht.

Navigationstopologie – Nach Themen angeordnete Struktur von WCM-Objekten innerhalb einer Website (Themenstruktur)

Objektdaten – Bestandteile eines WCM-Objekts: Inhalt, Metadaten und Protokoll

Objektkategorie – Zuordnung eines WCM-Objekts zu einer bestimmten Kategorie. Aufgrund dieser Zuordnung hat das WCM-Objekt eine Reihe von zusätzlichen Spezialattributen (Metadaten).

Objekttyp – Bezeichnet die spezifische Art des jeweiligen Objekts, z.B. “HTML-Seite”, “HTML-Vorlage”, “Thema”. Aus dem Objekttyp ergeben sich verschiedene Eigenschaften des WCM-Objekts. Der Objekttyp wird beim Anlegen des Objekts festgelegt und kann für einige Objekttypen nachträglich geändert werden. Objekttypen können über den Admin-Client oder den Content-Client bearbeitet werden.

Produktionssicht – Die Produktionssicht von Livelink WCM Server stellt die freigegebenen Seiten einer Website bereit. Mithilfe eines Webservers kann auf die Seiten über das Internet, Intranet oder Extranet zugegriffen werden.

QS-Sicht – Die QS-Sicht von Livelink WCM Server dient der Qualitätssicherung der Objekte und damit der Website-Inhalte. Diese Sicht stellt also die Kontrollinstanz zwischen der Bearbeitung in der Edit-Sicht und der Veröffentlichung in der Produktionssicht dar.

Session – Einheit, die von der JSP-Engine verwaltet wird, damit logisch zusammenhängende Aktionen (aus Sicht der Ressourcen) auch zusammengefasst werden können

SOAP – Simple Object Access Protocol. Standard, der den plattformunabhängigen Zugriff auf Web-Services ermöglicht bzw. die Datenübertragung zwischen dem Dienstanbieter und dem Dienstnehmer der Web-Services definiert. Das verwendete Austauschformat ist XML.

Status – Der Bearbeitungszustand eines WCM-Objekts. Änderungen des Status werden durch entsprechende Aktionen am WCM-Objekt veranlasst.

Synchrone Aktion – Eine synchrone Aktion kehrt erst zurück, wenn alle damit verbundenen Aufgaben abgeschlossen sind. Siehe auch *asynchrone Aktion*.

Topologie – Die hierarchische Anordnung von WCM-Objekten nach bestimmten Kriterien. Neben der *Navigationstopologie* (Anordnung nach Themen mit ihren jeweiligen Unterobjekten) gibt es auch eine *Vorlagentopologie*.

Transaktion – Durch eine Transaktion können mehrere Aktionen miteinander verbunden werden, sodass alle Änderungen erst wirksam werden, wenn die Transaktion explizit beendet wird.

UDDI – Universal Description, Discovery and Integration. Dieser XML-basierte Standard legt fest, wie Detailinformationen zu Web-Services und deren Anbietern in einheitlicher Form in Verzeichnissen abgelegt werden können.

VIPP – VIP Protocol. Proprietäres Protokoll zum Austausch von Daten zwischen den Komponenten eines WCM-Systems. Für die Kommunikation in WANs oder über das Internet kann VIPP in HTTP getunnelt werden.

Vorlagentopologie – Nach Vorlagen angeordnete Struktur von WCM-Objekten innerhalb einer Website (Vorlagenstruktur)

Vorlegen – Staging-Aktion von Livelink WCM Server: Bevor ein neu angelegtes oder geändertes Objekt veröffentlicht werden kann, muss es der Qualitätssicherung zur Prüfung vorgelegt werden. Durch das Vorlegen werden die Änderungen am Objekt in der QS-Sicht sichtbar.

Web-Services – XML-basierter Standard für Schnittstellen. Web-Services ermöglichen eine direkte Kommunikation zwischen Programmen, die in unterschiedlichen Programmiersprachen geschrieben werden und auf verschiedenen Plattformen laufen, über das standardisierte Protokoll SOAP. Ein weiterer großer Vorteil von Web-Services ist die Verwendung von Standard-Web-Formaten und -Protokollen: XML, HTTP und TCP/IP.

Workflow-Definition – Beschreibung eines Workflows. Eine Workflow-Definition setzt sich aus Start- und Endpunkten sowie Aktivitäten und Übergängen zusammen. Workflow-Definitionen werden im Format XPDL gespeichert und können über den Content Workflow Modeler grafisch bearbeitet werden.

WSDL – Web Services Description Language. Beschreibungssprache für Web-Services auf der Grundlage von XML. Für die Nutzung eines Web-Service durch eine Applikation ist eine genaue "Sprachregelung" bzw. Beschreibung erforderlich. Die Beschreibung muss genaue Informationen liefern über das verwendete Protokoll, die Adresse und Port-Nummer, die möglichen Prozeduren und Funktionen sowie die Formate für Input und Output. Der Anbieter eines Web-Service stellt diese Informationen in Form einer WSDL-Datei zur Verfügung.

XML – Extensible Markup Language. Ein Standard, der zur Normierung des formalen Aufbaus von Dokumenten verwendet wird.

XSL – Extensible Stylesheet Language. Ein Standard für Anweisungen, mit dem XML-Dokumente in andere Formate umgewandelt werden können. XSL wird häufig zur Umwandlung von XML in HTML genutzt.

Zugriffssteuerungsliste – Für jedes WCM-Objekt können Benutzer, Gruppen, Rollen und Gruppenrollen festgelegt werden, die Zugriff auf dieses Objekt haben sollen. Die einzelnen Zugriffsrechte werden für jeden Zugriffsberechtigten separat festgelegt. Wird auch als ACL (Access Control List) bezeichnet.

Zurückgeben – Staging-Aktion von Livelink WCM Server: Ein ausgeliehenes und bearbeitetes Objekt wird durch die Aktion "Zurückgeben" an das WCM-System zurückgegeben. Damit werden die vorgenommenen Änderungen in der Edit-Sicht sichtbar. Das Objekt wird wieder mit der Vorlage verknüpft und steht zur weiteren Bearbeitung zur Verfügung.

–

Index

A

Ablehnen 104
Abmelden 72
Acl 90
ACL ändern 91
AclEntry 91
Activity 96
addRemark 121
Aktivität 96
Alte Versionen eines WCM-Objekts zurückholen 145
Ändern
  ACL 91
Ändern von Metadaten 122
ASP.NET
  Beispiel 176
  ItemTemplate 186
  Page_Load 185
assignWorkflow 109
assignWorkflowAsync 110
Attribute zu einem WCM-Objekt abfragen 133
Attributmengen abfragen 65
Ausleihen 102
Ausleihen rückgängig machen 108
Authentifizierung 179
  C# 184
  SOAP-Toolkit 165

B

Bearbeiten
  ACL 91
Beispiel
  ASP.NET 176
  C# 176
  Maildialog 182
  VisualBasic 156
Bemerkung zu Protokoll hinzufügen 121
Benutzerdaten abfragen 60

C

C#
  Beispiel 176
  Cookies 184
  filter 185
change 122
changePassword 71
checkIn 101
checkOut 102
checkReferencesForDelete 123
checkReferencesForRelease 124
checkReferencesForSubmit 125
convertContent 126
Cookies
  C# 184
copy 127
create 128

D

Datentyp
  Acl 90
  AclEntry 91
  Activity 96
  DeploymentWaitInfo 98
  EMailInfo 99
  MultiImportPart 92
  ObjectData 78
  ObjectId 77

ObjectState 86
ObjectType 84
Permission 88
Transition 96
Workflow 94
WorkflowNavigationInfo 97
Datentypen
allgemein 76
delete 129
Deployment
Aufgaben abfragen 137
Deploymentinhalt eines WCM-Objekts abfragen 134
Deploymentsysteme 98
abfragen 67
DeploymentWaitInfo 98
depublishPage 130
destroy 131
directRelease 103, 188
direkte Freigabe 103
DOM 170

E

Eltern-Objekte eines WCM-Objekts abfragen 139
E-Mail
Daten 99
EMailInfo 99, 188
Erzeugen
Seite 132
WCM-Objekte 128
externe Referenzen eines WCM-Objekts abfragen 138

F

Fehlerbehandlung 169
C# 190
SOAP-Beispiel 191
filter 151, 170
Filtern von WCM-Objekten 151, 171, 185
forward 112
Freigeben 105
Funktionsbereiche abfragen 64

G

generatePage 132
get 133
getAssignedJobs 113
getAssignedWorkflow 113
getAssignedWorkflowActivities 114
getAssignedWorkflows 114
getAttributeSets 65
getAvailableWorkflows 115
getCheckOutContent 134
getChildren 135
getContent 136
getDeploymentJobs 137
getDeploymentSystems 67
getExternalLinks 138
getFunctionalAreas 64
getGroupProfiles 58
getLanguages 70
getLastLogEntries 138
getObjectCategories 65
getObjectTypes 65
getParent 139
getRoleProfiles 59
getRunlevel 69
getUserProfiles 60
getVersionList 140
getVIPVersion 70
getWebsiteNames 66
getWorkflow 115
Gruppendaten abfragen 58

I

Import verknüpfter Objekte 142
Inhalt eines WCM-Objekts erhalten 136
isForwardable 116
isRemovable 116
ItemTemplate 186

K

Kind-Objekte eines WCM-Objekts abfragen 135
Kopieren 127

L

listAllRunlevels 70
logout 72
Löschen 129
Löschen endgültig 131
Löschen Seite aus Dateisystem 130

M

Metadaten 78
Methoden
- addRemark 121
- assignWorkflow 109
- assignWorkflowAsync 110
- change 122
- changePassword 71
- checkIn 101
- checkOut 102
- checkReferencesForDelete 123
- checkReferencesForRelease 124
- checkReferencesForSubmit 125
- convertContent 126
- copy 127
- create 128
- delete 129
- depublishPage 130
- destroy 131
- directRelease 103, 188
- filter 151
- forward 112
- generatePage 132
- get 133
- getAssignedJobs 113
- getAssignedWorkflow 113
- getAssignedWorkflowActivities 114
- getAssignedWorkflows 114
- getAttributeSets 65
- getAvailableWorkflows 115
- getCheckOutContent 134
- getChildren 135
- getContent 136
- getDeploymentJobs 137
- getDeploymentSystems 67
- getExternalLinks 138
- getFunctionalAreas 64
- getGroupProfiles 58
- getLanguages 70
- getLastLogEntries 138
- getObjectCategories 65
- getObjectTypes 65
- getParent 139
- getRoleProfiles 59
- getRunlevel 69
- getUserProfiles 60
- getVersionList 140
- getVIPVersion 70
- getWebsiteNames 66
- getWorkflow 115
- isForwardable 116
- isRemovable 116
- listAllRunlevels 70
- logout 72
- move 141
- multiImport 142
- reject 104, 188
- release 105, 188
- removeWorkflow 117
- removeWorkflowAsync 118
- restoreVersion 145
- sortParentsFirst 146

submit 106, 188
submitImmediately 107
substituteLogin 72
undoCheckout 108
move 141
multiImport 142
MultiImport-Info 92
MultiImportPart 92

O

ObjectData 78
ObjectId 77
ObjectState 86
ObjectType 84
Objektkategorien abfragen 65
Objektserialisierung 158
Objekttyp 84
abfragen 65
OID 77

P

Passwort ändern 71
Permission 88
Protokolleinträge eines WCM-Objekts abfragen 138

R

reject 104, 188
release 105, 188
removeWorkflow 117
removeWorkflowAsync 118
restoreVersion 145
Rollendaten abfragen 59
Runlevel
aktuellen abfragen 69
alle abfragen 70

S

Session-Timeout 184
Session-Verwaltung
C# 184
SOAP-Toolkit 166
SOAP
Beispiel 156
Body 156
Envelope 156
SOAP-Toolkit 162
Authentifizierung 165
Verbindungsaufbau 165
sortParentsFirst 146
Sprachen abfragen 70
submit 106, 188
submitImmediately 107
substituteLogin 72
Suchen nach WCM-Objekten 151

T

Timeout 184
Transition 96

U

Übergang 96
undoCheckout 108

V

Verbindungsaufbau
C# 184
Verbundobjekt konvertieren 126
Verschieben 141
Version 193
Version abfragen 70
Version zurückholen 145
Versionsliste eines WCM-Objekts abfragen 140
Vertreter anmelden 72

VisualBasic 156
VisualStudio 176
    Projekt einrichten 177
    Webverweis 177
Vorlegen 106, 107

W

WCM-Objekt
    Attribute 78
Web-Services
    allgemein 19
Websitenamen abfragen 66
Weiterleiten 112
Workflow 94
    entfernen 117
    zuordnen 109
WorkflowNavigationInfo 97
WSDL
    Nutzung in VisualStudio 177

X

XML
    Serialisierung 158
XML-Serialisierung
    Beispiel 169
    href 170
    multiRef 170
XPath 170

Z

Zeitbegrenzung 184
Zerstören 131
Zugriffsrecht 88
Zugriffssteuerungsliste 90
Zugriffssteuerungsliste ändern 91
Zurückgeben 101
Zustand eines WCM-Objekts 86